Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Advancements in The Simulation of High Granular Calorimeters for High Energy Physics using Generative Machine Learning Techniques

Dissertation
zur Erlangung des Doktorgrades
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Physik
der Universität Hamburg

vorgelegt von
William Korcari

Hamburg
2025

Gutachter/innen der Dissertation:	Prof. Dr. Gregor Kasieczka Prof. Dr. Peter Schleper
Zusammensetzung der Prüfungskommission:	Prof. Dr. Daniela Pfannküche Prof. Dr. Gregor Kasieczka Prof. Dr. Peter Schleper Dr. Frank Gaede Prof. Dr. Oliver Gerberding
Vorsitzende/r der Prüfungskommission:	Prof. Dr. Daniela Pfannküche
Datum der Disputation:	16.06.2025
Vorsitzender des Fach-Promotionsausschusses PHYSIK:	Prof. Dr. Markus Drescher
Leiter des Fachbereichs PHYSIK:	Prof. Dr. Wolfgang J. Parak
Dekan der Fakultät MIN:	Prof. Dr.-Ing. Norbert Ritter

Eidesstattliche Versicherung / Declaration on oath

Hiermit versichere ich an Eides statt, die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt zu haben.

Sofern im Zuge der Erstellung der vorliegenden Dissertationsschrift generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Hamburg, den 24.03.2025

Unterschrift der Doktorandin/des Doktoranden

Zusammenfassung

Der Bereich der Teilchenphysik stützt sich bei seinen Analysen in hohem Maße auf simulierte Daten. Die zunehmende Menge an Messdaten führt dazu, dass mehr Simulationen benötigt werden, um unser aktuelles Wissen über die Natur mit den tatsächlichen Messungen zu vergleichen. Ein Grund für den drastischen Anstieg der gemessenen Daten in der Zukunft ist das High Luminosity Upgrade am LHC, bei dem Kollisionen mit einer viel höheren Rate stattfinden werden, wodurch die Anzahl der gemessenen Kollisionen drastisch ansteigt.

Von besonderem Interesse für den Umfang dieser Arbeit ist das CMS High Granular Calorimeter (HGCal), das die derzeitigen Endkappen-Kalorimeter von CMS ersetzen wird. Dieser Detektor umfasst ca. 3 Millionen hexagonale Auslesezellen pro Endkappe, was ihn zu einem Gerät macht, das in der Lage ist, feinkörnige Schauer zu erzeugen. Außerdem wird ein System implementiert, das den Zeitpunkt einer Messung in einer Detectorzelle mit einer Auflösung von etwa 30 ps aufzeichnen kann, was bei der Unterdrückung von pile-up und der Rekonstruktion von Trajektorien individueller Teilchen hilfreich sein wird. Generatives maschinelles Lernen hat in letzter Zeit an Bedeutung gewonnen, da sie das Potenzial hat, die Standardsimulationstechniken zu ergänzen..

Diese Arbeit konzentriert sich auf mehrere generative Modelle, die uns dem Ziel einer schnelleren und genaueren Simulation näher bringen. Die erste Studie wurde mit Graph Neural Networks durchgeführt, da Graphen eine sehr natürliche Art und Weise sind, elektromagnetische Schauer zu beschreiben, aber in Bezug auf die Skalierbarkeit Mängel aufweisen. Wir fanden heraus, dass es sinnvoll ist, bereits vorhandene Informationen wie die Geometrie des Kalorimeters zu nutzen, um ein solches Netzwerkarchitektur Netz zu trainieren, aber die hohe Kardinalität führte uns in die Richtung von Graphen, die mit diesen Informationen wachsen können, aber nur bis zur benötigten Schauergröße, anstatt die gesamte Anzahl der jederzeit verfügbaren Zellen zu nutzen. Da sich dieser erste Versuch als zu schwierig erwies und sich die Technologie weiterentwickelte, haben wir das EPiC-GAN-Modell verwendet, das eine gute Genauigkeit und eine hohe Generierungsgeschwindigkeit bei Schauern mit geringerer Komplexität zeigte, aber nicht auf die Kardinalität des HGCal skalieren konnte. Schließlich haben wir CaloClouds II implementiert, ein Modell, das eine Kombination aus einem Diffusion Modell mit kontinuierlicher Zeit und einem Normalizing Flow ist, um nicht nur das HGCal erfolgreich simulieren zu können, sondern auch die Time-of-Hits-Funktion einzubeziehen, die eine entscheidende Integration in dieses Detektor-Upgrade sein wird.

Abstract

The field of Particle Physics heavily relies on simulated data in order to perform analyses. The increase in the amount of measured data translates in the need for more simulations used to compare out current knowledge of Nature to actual measurements. One reason for a drastic increase of measured data in the future is the High Luminosity upgrade at the LHC, which will feature collisions at a much higher rate thus drastically increasing the number of measured collisions.

Of particular interest for the scope of this work is the CMS High Granular calorimeter (HGCal), which will replace CMS's current endcap calorimeters. This detector comprises circa 3 million readout hexagonal cells per endcap, making it a machine capable of producing fine-grained showers. It will also implement a system capable of recording the time of a hit measurement with a resolution of circa 30 ps, which will help with pile-up rejection and track reconstruction. Generative Machine Learning has risen recently as it has the potential to augment standard simulation techniques.

This thesis focuses on multiple generative models that bring us closer to the goal of faster and more accurate simulation. The first study was performed on Graph Neural Networks, as graphs are a very natural way to describe electromagnetic showers, but this model architecture lacks in terms of scalability. We found that there is value in utilizing already given information like the geometry of the calorimeter to train such a network, but the high cardinality led us toward the direction of graphs that could grow using that information but only until the needed shower size instead of using the whole number of cells available at all times. As this first attempt proved to be too challenging and the technology evolved, we then moved on with the EPiC GAN model, which showed good fidelity and high generation speed on showers with reduced complexity but failed to scale up to the cardinality of the HGCal. Finally, we implemented CaloClouds II, a model that is a combination of a continuous-time diffusion model and normalizing flow, to not only be able to successfully simulate the HGCal calorimeter but to do so by also including the time-of-hits feature which will be a crucial integration in this detector upgrade.

Contents

1 Introduction **1**

2 High Energy Physics Theoretical Framework **5**
2.1 The Standard Model 5
2.2 Physics Beyond the Standard Model Physics 9
2.3 Collider Experiments 11
2.4 The CMS Experiment 14

3 Calorimetry **21**
3.1 Particles Interactions with Matter 21
3.2 Particle Showers 26
3.3 Calorimeters 28
3.4 The CMS High Granularity Calorimeter 29
3.5 Calorimeter simulation 33

4 Machine Learning **37**
4.1 Gradient Descent 37
4.2 Optimizers 38
4.3 Loss Functions 39
4.4 Evaluation Metrics 40
4.5 Machine Learning Challenges 42
4.6 Neural Networks 42
4.7 Conclusion 46

5 Generative Machine Learning **47**
5.1 Generative Adversarial Networks 47
5.2 GAN variants 49
5.3 Flow Models 50
5.4 Diffusion Models 53

6 Shared Data and Algorithms for Physics Data **57**
6.1 Datasets 58
6.2 Python Interface 61
6.3 Example Application 62
6.4 Outlook and Conclusions 68

7 Dynamic Graph Neural Networks for High-Granular Calorimeters **71**
7.1 Dataset 71
7.2 Dynamic Graph Convolutional Networks 72

8 EPiC GAN for 4-Dimensional Calorimeter Data **77**
8.1 Data Samples 77

8.2 EPiC GAN Architecture 78
8.3 Results 80
8.4 Conclusions 84

9 CaloClouds for the CMS HGCal **85**
9.1 Data: format and processing 86
9.2 Generative Model 86
9.3 Projection to Geometry 87
9.4 Results 88
9.5 Conclusions 95

10 Conclusion **97**

Bibliography **101**

1 Introduction

The high-energy physics (HEP) field is driven by a fundamental quest to understand the nature of the universe at its most elementary level. Particle physics experiments, particularly those conducted at large collider facilities such as the Large Hadron Collider (LHC) at CERN, have provided profound insights into the fundamental constituents of matter and the forces governing their interactions. These experiments rely on highly sophisticated detector systems designed to capture and analyze the vast amounts of data generated by high-energy particle collisions. One of the critical components of these detectors is the calorimeter, an instrument used to measure the energy of particles through their interactions with matter. Calorimetry plays an essential role in reconstructing the properties of fundamental particles and identifying signatures of new physics beyond the Standard Model.

In recent years, the increasing complexity and scale of high-energy physics experiments have led to a growing reliance on advanced computational techniques, including machine learning (ML), to enhance data analysis and simulation processes needed reproduce our state-of-the-art knowledge to the Physics theory and compare it to our measurements of Nature. Traditional simulation methods based on Monte Carlo techniques, while highly accurate, are computationally expensive and require significant resources to achieve the necessary levels of precision. The advent of machine learning methods has introduced a paradigm shift in the way calorimeter simulations are performed, offering potential solutions to the challenges of computational efficiency while maintaining high fidelity. The imminent improvement of the LHC with the High-Luminosity upgrade will cause the luminosity to increase and correspondingly the amount of simulated data will need to scale up, posing a very tough challenge [1]. Figure 1.1 illustrates the fraction of CPU resources that will be allocated to the various necessary tasks if no R&D comes into play, while Figure 1.2 shows the increment of CPU time needed in comparison to the estimated increment of available CPU with and without R&D. This dissertation explores the application of machine learning techniques to the simulation of high-granular calorimeters, particularly within the context of the Compact Muon Solenoid (CMS) experiment at the LHC. In particular, the endcap calorimeters of CMS will be substituted with the new CMS High-Granularity Calorimeter (HGCal) [2]. The CMS HGCal represents a significant advancement in detector technology. It features a high level of segmentation in both lateral and longitudinal directions, enabling precise spatial and temporal measurements of particle showers. This increased granularity allows for improved pile-up mitigation, better energy resolution, and enhanced particle identification capabilities. However, the computational cost associated with simulating such a complex calorimeter is substantial. The traditional approach, based on Monte Carlo methods such as those implemented in the Geant4 simulation toolkit, requires extensive computing resources to accurately model the interactions of particles within the detector. As experimental datasets grow in size and complexity, the demand for faster and more efficient simulation methods

has become increasingly pressing.

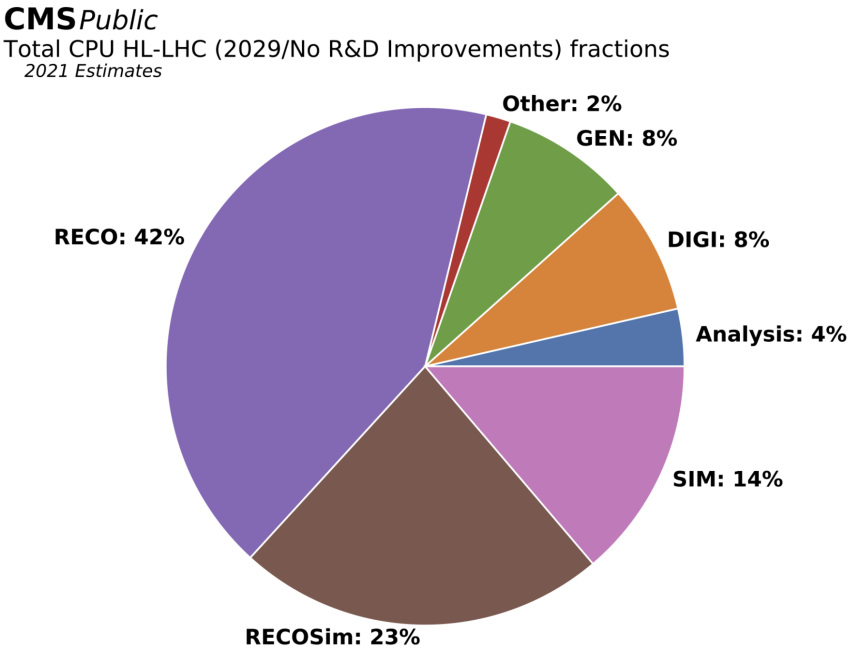


Figure 1.1: Breakdown of CPU usage without any R&D improvement [3].

Machine learning techniques offer promising solutions to the challenges of calorimeter simulation. Deep learning models, including GANs, normalizing flows, and diffusion models, have demonstrated remarkable success in generating realistic calorimeter showers with significantly reduced computational requirements compared to traditional methods. These models learn to approximate the complex distributions of particle interactions within the calorimeter by training on large datasets of simulated events. Once trained, they can generate new samples with high fidelity at a fraction of the computational cost, making them an attractive alternative for large-scale simulations in high-energy physics.

This dissertation investigates several machine learning approaches to calorimeter simulation, with a particular focus on generative models. GANs have emerged as powerful tools for generating synthetic data that closely resemble real detector responses. A GAN consists of two neural networks—a generator and a discriminator—which are trained in a competitive framework. The generator learns to produce realistic samples, while the discriminator attempts to distinguish between real and generated data. Through this adversarial process, the generator improves its ability to generate high-quality calorimeter showers that mimic those produced by Monte Carlo simulations.

Another class of generative models explored in this work is normalizing flow models. These models learn a bijective mapping between a simple prior distribution and the complex target distribution of calorimeter showers. By leveraging invertible transformations, normalizing flows enable efficient sampling and density estimation, making them well-suited for applications in high-energy physics. Diffusion models, inspired by non-equilibrium thermodynamics, have also gained traction as a robust generative modeling approach. These models simulate the gradual transformation of noise into structured calorimeter showers through a series of stochastic steps, offering high-quality sample generation with strong theoretical guarantees.

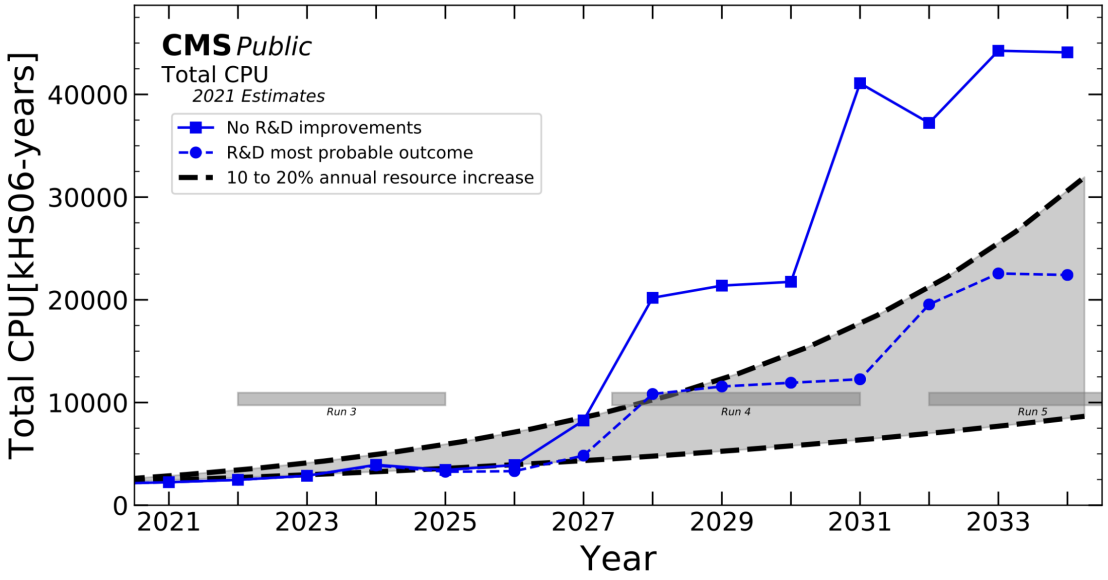


Figure 1.2: Projections of needed CPU. On the plot, the gray band represents the Baseline and Weighted Probable (dashed line). The effect of GPUs is not represented in this plot. In the legend, the Baseline scenario is described as "No R&D improvement" and the Weighted Probable scenario as "R&D most probable outcome". Taken from Ref. [3].

Beyond generative models, this dissertation also examines the application of graph neural networks (GNNs) to calorimeter data analysis. Given the highly granular nature of modern calorimeters, representing particle showers as graph structures allows for more flexible and efficient data processing. GNNs leverage the relational structure of calorimeter hits to learn spatial correlations and improve event reconstruction. Dynamic graph convolutional networks and point cloud neural networks are explored as potential architectures for extracting meaningful features from calorimeter data.

A key contribution of this work is the development of the EPiC GAN, a novel generative model designed for four-dimensional calorimeter data. This model integrates energy, spatial, and timing information to produce high-fidelity calorimeter simulations. The effectiveness of the EPiC GAN is evaluated through a series of benchmark tests comparing its performance against traditional Monte Carlo methods and other generative models. Results demonstrate that the EPiC GAN achieves state-of-the-art performance in generating realistic calorimeter showers while significantly reducing computational costs.

Additionally, this dissertation presents the CaloClouds II model, a generative approach that projects simulated calorimeter showers onto the detector geometry. This method enhances the interpretability of generated showers and facilitates direct comparisons with experimental data. By incorporating geometric constraints and physical priors into the generative process, CaloClouds improves the consistency and accuracy of machine learning-based calorimeter simulations.

The structure of this dissertation is as follows: Chapter 2 provides a theoretical overview of high-energy physics, including the Standard Model, beyond Standard Model physics, and collider experiments. Chapter 3 introduces the fundamentals of

calorimetry, describing particle interactions with matter and the design of modern calorimeters. Chapter 4 presents an overview of machine learning techniques relevant to calorimeter simulation, while Chapter 5 delves into generative machine learning models, including GANs, normalizing flows, and diffusion models. Chapter 6 shows the contribution of shared data and architectures for physics. Chapter 7 explores the application of graph neural networks to calorimeter data, and Chapter 8 introduces the EPiC GAN model. Chapter 9 discusses the CaloClouds II applied to the CMS High Granular Calorimeter, and Chapter 10 concludes with a summary of the explored work.

2 High Energy Physics Theoretical Framework

The Standard Model (SM) of particle physics is the most successful theories to describe nature at the fundamental level. It is a quantum field theory that describes the electromagnetic, weak, and strong interactions. While the SM has been tested in a wide range of experiments and has been able to predict the outcome of many of them, it doesn't describe Nature in its entirety, as it does not include an explanation for gravity, dark matter, and dark energy, among the others. In this chapter, several aspects of the SM that are relevant to this thesis will be described. That includes the Higgs mechanism, which is responsible for the generation of mass of the elementary particles. This chapter is meant as a theoretical background for the work and results presented in this thesis.

2.1 The Standard Model

The Standard Model [4–7] is a quantum field theory that describes the electromagnetic, weak and strong interactions. The SM is based on the gauge symmetry group $SU(3)_C \times SU(2)_L \times U(1)_Y$ [8], where $SU(3)_C$ is the symmetry group of the strong interaction mediated by gluons, while the $SU(2)_L \times U(1)_Y$ is the symmetry group of the electroweak interaction which is mediated by the photon, the W and Z bosons.

The fermionic content of the SM is depicted in the leftmost three columns of Figure 2.1. These fermions can be categorized into leptons which consist of charged leptons (ℓ^-) and lepton neutrinos (ν_ℓ) that have spin 1/2, while quarks can be classified as up-type quarks (q_u) and down-type quarks (q_d). Charged leptons and quarks carry an electromagnetic charge: leptons have a charge of -1, up-type quarks have a charge of $\frac{2}{3}$, and down-type quarks have a charge of $-\frac{1}{3}$. Additionally, quarks possess a color charge, associated with the strong interaction.

Leptons and quarks are organized into three generations, which differ in mass and flavor but share identical charge and spin. For charged leptons, the three generations are electrons (e), muons (μ), and taus (τ), while for neutral leptons they are electron-neutrinos (ν_e), muon-neutrinos (ν_μ), and tau-neutrinos (ν_τ). The up-type quarks consist of the up (u), charm (c), and top (t) quarks, whereas the down-type quarks include the down (d), strange (s), and bottom (b) quarks. Each fermion has a corresponding antiparticle with opposite charge and flavor.

Fermions in each generation can be decomposed into left- and right-handed chirality components as follows:

$$\nu_\ell, q_u, \ell^-, q_d = \begin{pmatrix} \nu_\ell \\ \ell^- \end{pmatrix}_L, \begin{pmatrix} q_u \\ q_d \end{pmatrix}_L, \ell^-_R, q_{u_R}, q_{d_R}, \tag{2.1}$$

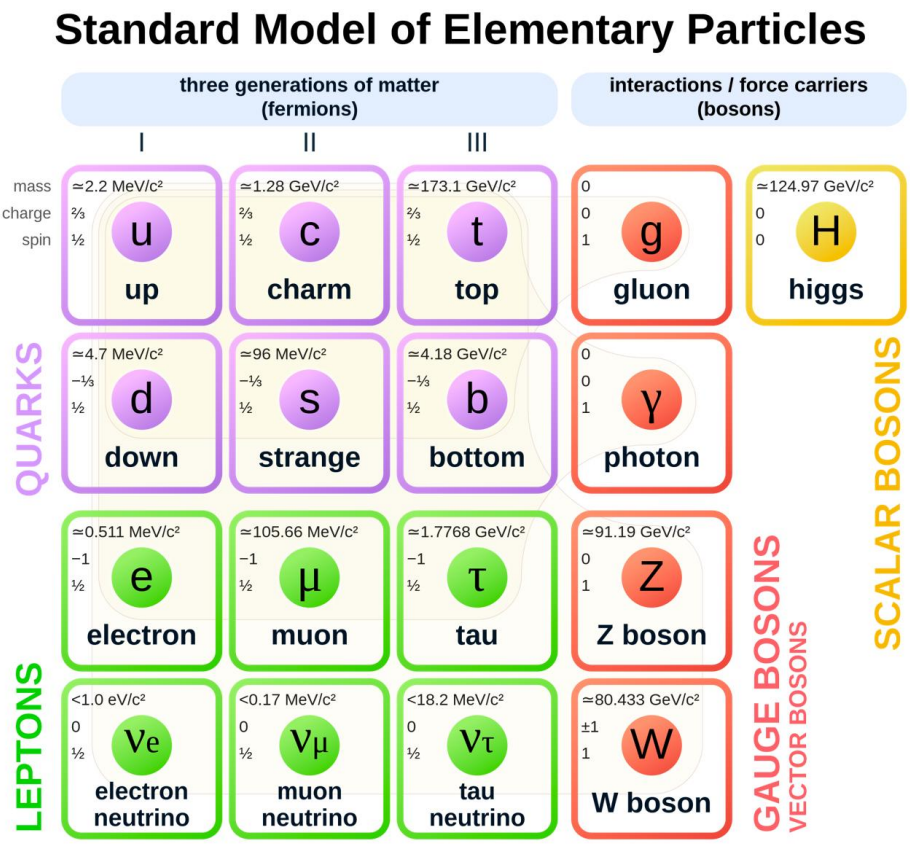


Figure 2.1: The particles of the Standard Model, from Ref. [9]

where left-handed (L) fermions transform as $SU(2)_L$ doublets, and right-handed (R) fermions transform as $SU(2)_L$ singlets. In the Standard Model, the neutrino is considered massless, so there is no right-handed neutrino component (ν_{ℓ_R}) nor left-handed anti-neutrinos.

All left-handed particles possess weak isospin, the charge corresponding to the weak interaction.

Fundamental Forces

Each fundamental force in the SM is mediated by the exchange of bosons with integer spin. For electromagnetic interactions, described by quantum electrodynamics (QED), the mediator is the massless, chargeless photon (γ). Photons couple to the electromagnetic charge and therefore interact only with charged leptons, quarks, and charged bosons.

The weak interaction is mediated by three particles: the neutral Z boson and the charged W^+ and W^- bosons. These bosons interact exclusively with left-handed fermions, as right-handed fermions lack the required isospin. Notably, the weak interaction does not generally conserve flavor, allowing processes such as the decay of a t quark into a b quark and a W^+ boson.

The mediators of the strong force, described by quantum chromodynamics (QCD), are massless, electromagnetically neutral gluons. Gluons couple to the color charge and carry color charge themselves, enabling interactions both with quarks and other gluons. This gluon-gluon self-interaction leads to the phenomenon of *color confinement* [10], where free particles with a net color charge cannot exist. Consequently, color-carrying particles are always confined in composite states that are color-neutral. Common examples of such states are baryons, composed of three quarks (e.g. protons and neutrons), and mesons, which are quark-antiquark pairs (e.g. pions and kaons). Rare states such as tetraquarks [11] and pentaquarks [12] have also been observed. Another consequence of confinement is that quarks cannot propagate freely; instead, they produce a spray of color-neutral hadrons, leading to the phenomenon of particle jets.

Interactions between particles are governed by the Standard Model Lagrangian, which includes terms coupling the fields of the interacting particles that are represented by quantum fields. For instance, the interaction between a fermion-antifermion pair, $\bar{\psi}\psi$, and the photon gauge field A_μ is expressed as:

$$iq\bar{\psi}\gamma^\mu A_\mu \psi, \tag{2.2}$$

where q is the electromagnetic charge of ψ and determines the interaction strength.

The probability of a scattering event between two particles is characterized by its cross-section, σ. For a two-particle-to-two-particle interaction, the cross-section is given by:

$$\sigma = \frac{1}{64\pi^2 s}\frac{p_f}{p_i}\int |M_{fi}|^2 \, d\Omega, \tag{2.3}$$

as described in Reference [10]. Here, p_f and p_i denote the momenta of the final and initial states in the center-of-mass frame, respectively, s is the squared center-of-mass

energy, and M_{fi} is the matrix element for the transition between the initial and final states. The matrix element depends on the coupling constants of the involved interactions and may have an angular dependence, requiring integration over all possible angles Ω in spherical coordinates.

The angle-dependent differential cross-section can be expressed as:

$$\frac{d\sigma}{d\Omega} = \frac{1}{64\pi^2 s}\frac{p_f}{p_i}|M_{fi}|^2. \tag{2.4}$$

The Higgs Mechanism

The masses of gauge bosons cannot be directly included in the Standard Model Lagrangian, as doing so would violate gauge invariance, a fundamental principle of the theory [10]. Gauge invariance requires that the Lagrangian remains invariant under local transformations of the gauge group. Introducing explicit mass terms for gauge bosons would break this invariance, undermining the structure of the Standard Model.

Similarly, fermion masses cannot be introduced naively. The fermion mass term takes the form:

$$m\bar{\psi}\psi = m(\psi_R\bar{\psi}_L + \psi_L\bar{\psi}_R), \tag{2.5}$$

where ψ represents the fermion field, ψ_R is the right-handed component, and ψ_L is the left-handed component. Right-handed fields transform as singlets under $SU(2)_L$, meaning they are unaffected by $SU(2)_L$ transformations, while left-handed fields transform as doublets. The mass term links these two components, thereby explicitly breaking the $SU(2)_L$ symmetry.

The Higgs mechanism [13–15] elegantly resolves these issues by introducing a complex scalar field, ϕ. This field is endowed with a potential of the form:

$$V = \mu^2\phi^\dagger\phi + \lambda(\phi^\dagger\phi)^2, \tag{2.6}$$

where μ^2 and λ are parameters of the potential which respects gauge invariance. However, when μ^2 is negative, the potential attains its minimum at a nonzero value of ϕ, i.e. $\phi \neq 0$. As a result, the scalar field develops a nonzero vacuum expectation value (vev), denoted by v. This spontaneous symmetry breaking leads to the generation of masses for the gauge bosons while preserving the gauge invariance of the underlying Lagrangian.

The masses of the gauge bosons arise through their interactions with the scalar field. The W bosons acquire a mass:

$$m_W = \frac{gv}{2}, \tag{2.7}$$

and the Z boson acquires a mass:

$$m_Z = \frac{\sqrt{g^2 + g'^2}\,v}{2}, \tag{2.8}$$

where g and g' are the electroweak coupling constants. Importantly, the photon remains massless, as required by the unbroken $U(1)_{\text{em}}$ gauge symmetry.

In the unitary gauge, for fluctuations around the minimum, the scalar field ϕ can be expressed as:

$$\phi = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v+h \end{pmatrix}, \tag{2.9}$$

where h represents fluctuations around the vacuum expectation value and corresponds to the physical Higgs boson. This formulation reveals that the scalar field not only generates masses for the gauge bosons but also introduces a new particle with spin 0, the Higgs boson, as a fundamental feature of the Standard Model.

In addition to generating gauge boson masses, the Higgs mechanism provides a framework for giving masses to fermions. Fermion masses are introduced through the Yukawa interaction, which takes the form:

$$g_f(\psi_R \phi \bar{\psi}_L + \psi_L \phi^\dagger \bar{\psi}_R), \tag{2.10}$$

where g_f is the Yukawa coupling constant specific to each fermion. Expanding this interaction yields:

$$g_f v(\psi_R \bar{\psi}_L + \psi_L \bar{\psi}_R) + g_f h(\psi_R \bar{\psi}_L + \psi_L \bar{\psi}_R). \tag{2.11}$$

The first term generates the fermion masses, with the relationship $g_f = \frac{m_f}{v}$, where m_f is the fermion mass. The second term describes the interaction between the fermions and the Higgs boson, with an interaction strength proportional to the fermion mass. This proportionality explains why heavier fermions have stronger couplings to the Higgs boson.

The Higgs mechanism, proposed in 1964, predicted the existence of the Higgs boson as a cornerstone of the Standard Model. In 2012, experiments at the ATLAS and CMS detectors at CERN confirmed this prediction with the discovery of a scalar particle consistent with the Higgs boson [16, 17]. Measurements of this particle's properties, including its decay channels and interaction strengths, align with theoretical predictions, providing robust evidence for the validity of the Higgs mechanism. This discovery marked a pivotal achievement in particle physics, completing the Standard Model's framework.

2.2 Physics Beyond the Standard Model Physics

Despite its success in describing fundamental particles and their interactions, the Standard Model (SM) has several significant limitations and unanswered questions, motivating the search for physics beyond the Standard Model (BSM). Below, we outline some key issues:

- **Gravity:** While the effects of gravity are well described on macroscopic scales by the theory of general relativity [18], its unification with the SM remains one of the unsolved problems in physics as this fundamental force is currently completely absent from the SM. The Planck scale, where quantum gravitational effects become significant, is approximately at 10^{18} GeV, far beyond the reach of current experiments. The search for a quantum theory of gravity is a central goal of theoretical physics. String theory [19] and loop quantum gravity [20] are two prominent candidates for a quantum theory of gravity.

- **Dark Matter and Dark Energy:** Observations of gravitational effects, such as the rotation curves of galaxies and galaxy clusters, indicate that ordinary baryonic matter accounts for only about 5% of the universe's total mass-energy. Approximately 25% is attributed to dark matter, which interacts gravitationally but does not emit or absorb light. The remaining 70% is associated with dark energy, which appears as a constant energy density in the equations of general relativity. So, overall, the SM does not account for more than 90% of the energy in the universe. In some theories, dark matter is explained as new particles that interact weakly with the SM. One example is Weakly Interacting Massive Particles (WIMPs), which could potentially be discovered in high-energy collider experiments.
- **Neutrino Masses:** The SM does not provide a mechanism for neutrinos to have mass. However, the phenomenon of neutrino oscillations, where neutrinos change flavor as they propagate, has been experimentally observed and requires neutrinos to possess mass. This discovery was confirmed by experiments such as Super-Kamiokande [21] and the Sudbury Neutrino Observatory [22], for which the 2015 Nobel Prize in Physics was awarded. Recently, the KATRIN experiment established an upper limit on the mass of the electron antineutrino of $m < 0.45\,\mathrm{eV}$ at 90% confidence level [23].
- **Matter-Antimatter Asymmetry:** The Big Bang theory suggests that equal amounts of matter and antimatter were created in the early universe. However, the observable universe is dominated by matter, with very little antimatter present. The SM does not explain this baryon asymmetry. The imbalance between matter and antimatter is one of the most important questions that remain unanswered in modern physics. The problem arises from the natural assumption that the universe is neutral to begin with. The observed asymmetry of matter and antimatter in the universe is not accounted for in the SM. If that were the case, an equal amount of left-handed baryons and right-handed antibaryons would be produced. The SM contains a charge parity (CP)-violating process, as theoretized [24, 25] and first observed in Ref. [26], due to the complex phase present in the Cabibbo-Kobayashi-Maskawa (CKM) flavor mixing matrix. The CKM matrix describes the mixing between the mass eigenstates of quarks under the weak interaction. This complex phase explains the CP-violating effects observed in experiments involving neutral kaons and B mesons. However, the amount of CP violation predicted by the SM is not enough to account for the observed baryon asymmetry in the universe.
- **The Hierarchy Problem:** Without fine-tuned quantum corrections, the Higgs boson mass would naturally rise to the Planck scale ($\sim 10^{18}\,\mathrm{GeV}$) due to higher order contributions. Such a high mass would also affect the masses of other SM particles, including quarks, leptons, and the $W^{\pm}$ and Z^0 bosons. The observed Higgs mass, however, is at the electroweak scale ($\sim 100\,\mathrm{GeV}$). This discrepancy, known as the hierarchy problem, challenges the principle of naturalness [27]. Supersymmetry [28] offers a potential solution by introducing new bosonic and fermionic partners to the SM particles whose quantum corrections cancel out. However, no experimental evidence for supersymmetric particles has been found to date.
- **Force Unification:** The unification of electromagnetism and the weak interac-

tion into the electroweak theory suggests the possibility of further unifying the electroweak and strong interactions. Such a framework is referred to as a Grand Unified Theory (GUT). GUTs often propose unification of quarks and leptons in representations such as $SU(5)$ [29], but experimental evidence is still lacking.

- **Ad-Hoc Parameters:** The SM requires 19 free parameters, including particle masses, CKM mixing angles, and gauge coupling constants, which are not derived from first principles and must instead be experimentally measured. Additionally, the SM does not explain why there are exactly three generations of leptons and quarks.

The search for BSM physics is a central focus of the high-energy physics community. Experiments at the Large Hadron Collider (LHC) at CERN, as well as future colliders such as the High-Luminosity LHC (HL-LHC) and the International Linear Collider (ILC), aim to explore energy scales beyond the reach of current experiments. The discovery of new particles or interactions would provide crucial insights into the fundamental nature of the universe and guide the development of new theoretical frameworks.

2.3 Collider Experiments

The SM of particle physics has been tested extensively in collider experiments, which provide a controlled environment for studying the interactions of fundamental particles. Colliders accelerate particles to high energies and collide them at specific interaction points, allowing researchers to probe the properties of matter at the smallest scales. In this section, we describe the key features of collider experiments and the detectors used to study the resulting collisions.

The Large Hadron Collider (LHC)

The **Large Hadron Collider (LHC)**, located at CERN, is currently the most powerful particle collider in the world [30]. This circular accelerator is capable of colliding protons, lead nuclei (Pb-Pb), or lead nuclei with protons (Pb-proton) at a maximum center-of-mass energy of 14 TeV. Since its commissioning, the LHC has been very impactful in advancing particle physics, with its most notable achievement being the discovery of the Higgs boson in 2012.

Modern particle colliders operate based on two essential components: **magnets** and **radio-frequency (RF) cavities**. The RF cavities are used to accelerate charged particles by imparting energy to them, while the magnets — comprising dipoles, quadrupoles, and higher-order multipole magnets—are responsible for steering and focusing the particle beams.

In a **circular collider** or **synchrotron**, such as the LHC, these components are arranged to create a closed, circular path. The circular configuration allows the particle beams to repeatedly pass through the RF cavities, gaining energy with each cycle. However, charged particles traveling along a curved path lose energy through the emission of radiation. This phenomenon, known as **Bremsstrahlung**, is referred to as **synchrotron radiation** when it occurs in circular accelerators. The energy loss

due to synchrotron radiation depends on three factors: the particle's velocity, the radius of the circular path, and the particle's mass.

Challenges and Solutions for Synchrotron Radiation

The energy loss due to synchrotron radiation imposes a practical limit on the energy that circular colliders can achieve. As particles approach higher velocities, the radiative losses increase, eventually balancing the energy gained from the accelerator. Two strategies help mitigate this limitation:

- **Increasing the bending radius:** A larger circular path reduces the curvature, leading to lower synchrotron radiation losses. Consequently, larger colliders can achieve higher beam energies.
- **Using heavier particles:** For particles of equal energy, heavier particles have lower velocities and a smaller Bremsstrahlung cross-section. Therefore, synchrotron radiation losses are significantly reduced for heavier particles.

This reasoning explains why the LHC accelerates protons and lead ions rather than electrons. Its predecessor, the Large Electron-Positron Collider (LEP), operated in the same tunnel with the same radius but was limited to lower energies due to the much greater synchrotron radiation losses for electrons.

Key Features of the LHC

The LHC consists of two beam pipes, each carrying a beam of particles circulating in opposite directions. The beams are divided into a maximum of 2808 **bunches**, with each bunch containing approximately 10^{11} particles. These beams are pre-accelerated through a series of smaller accelerators before being injected into the LHC at an initial energy of 450 GeV. The beams are then further accelerated to their final energy inside the LHC.

At four specific locations around the collider ring, the beams are crossed and brought into collision. These interaction points correspond to the sites of the four major LHC experiments:

- **ATLAS** [31] and **CMS** [32], which are general-purpose detectors designed to explore a wide range of particle physics phenomena.
- **LHCb** [33], which focuses on the physics of **b-quarks** and precision measurements of CP violation.
- **ALICE** [34], which specializes in studying heavy-ion collisions to investigate the quark-gluon plasma.

Luminosity: A Measure of Collider Performance

The performance of a collider is characterized not only by its energy but also by its **luminosity**, which measures the capability of the setup to bring the particles into the

relevant region for data taking. The **instantaneous luminosity** L is given by:

$$L = \frac{N_b^2 k_b f}{A} H_D, \tag{2.12}$$

where:

- N_b is the number of particles per bunch,
- k_b is the total number of bunches in the collider,
- f is the bunch crossing frequency,
- A is the effective area where the beams overlap, and
- H_D is a correction factor accounting for beam dynamics, such as bunch widening due to the electromagnetic fields of the particles.

The design luminosity of the LHC was initially set at $L = 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$. The ongoing **High-Luminosity LHC** upgrade (see next paragraph for more details) aims to increase this value by a factor of approximately five by focusing on reducing the beam overlap area, thus allowing for more frequent particle collisions.

In addition to instantaneous luminosity, the **integrated luminosity** L_{int} is a key parameter as it is used as a measure of the dataset size. It represents the total number of possible collisions over a given period and is defined as:

$$L_{\mathrm{int}} = \int L(t)\, dt. \tag{2.13}$$

The integrated luminosity is essential for calculating the expected number of occurrences N_{proc} of a particular particle physics process in a given timeframe with cross-section σ_{proc}:

$$N_{\mathrm{proc}} = L_{\mathrm{int}} \sigma_{\mathrm{proc}}. \tag{2.14}$$

The High Luminosity Upgrade at the LHC

The High Luminosity LHC (HL-LHC) project aims to significantly enhance the performance of the Large Hadron Collider (LHC) at CERN, boosting its potential for groundbreaking discoveries in particle physics. The primary objectives of the HL-LHC are to achieve a peak luminosity of $5 \times 10^{34}\ cm^{-2}s^{-1}$ with leveling and to accumulate an integrated luminosity of 250 fb^{-1} per year. The ultimate goal is to gather 3000 fb^{-1} in approximately 12 years of operation after the upgrade. This substantial increase in luminosity, a tenfold increase compared to the LHC's initial 12 years, will provide scientists with a much larger dataset for analysis. The HL-LHC project also focuses on ensuring high operational efficiency until 2035, maximizing the scientific output over its lifespan. The increased luminosity will enable more precise measurements of fundamental particles, such as the Higgs boson, and will allow scientists to probe deeper into the mysteries of dark matter and dark energy. Furthermore, the HL-LHC will facilitate the search for new particles and phenomena beyond the Standard Model of particle physics, potentially revolutionizing our understanding of the universe.

Superconducting Magnets: Pushing the Limits of Focus

The HL-LHC will feature cutting-edge superconducting magnets with peak magnetic fields reaching a staggering 11-12 Tesla. These magnets are crucial for focusing the proton beams to an incredibly small size at the interaction points, thereby increasing the likelihood of collisions. The real breakthrough lies in the use of a novel superconducting material, Niobium-tin (Nb3Sn), which can sustain magnetic fields far beyond the capabilities of the Niobium-titanium (NbTi) used in the current LHC magnets. This advancement in magnet technology is a cornerstone of the HL-LHC's luminosity enhancement.

Beam Rotation: The Art of the Crab Cavity

The HL-LHC will employ compact superconducting crab cavities to ensure that the proton bunches collide head-on, maximizing the overlap and, hence, the number of collisions. These cavities generate transverse electric fields that give each bunch a slight longitudinal"kick" effectively rotating them for a perfect head-on collision. This ingenious technique not only boosts the peak luminosity but also allows for dynamic control of the luminous region size during the fill, optimizing the collision density throughout the process.

Beam Collimation: Precision Cleaning for Intensity

With the HL-LHC's increased beam intensity comes the critical challenge of managing beam losses. Stray particles can cause detrimental effects, from quenching the superconducting magnets to damaging sensitive detector components. To mitigate this, the HL-LHC will utilize advanced beam collimation systems. These systems act as highly precise "beam cleaners", intercepting and safely absorbing any particles that deviate from the designated path. This meticulous control of beam losses is essential for maintaining the integrity and efficiency of the accelerator.

Superconducting Links: Powering the Future with Efficiency

The HL-LHC's superconducting magnets require a tremendous amount of power. To deliver this power with minimal loss, the upgrade will utilize long, high-power superconducting links. These links are crafted from high-temperature superconductors, capable of carrying significantly more current than their conventional counterparts. By minimizing energy dissipation during power transmission, these superconducting links contribute to the overall efficiency and sustainability of the HL-LHC.

2.4 The CMS Experiment

The Compact Muon Solenoid detector (CMS) [35, 36] is one of the two general-purpose detectors at the Large Hadron Collider (LHC) at CERN. The CMS detector is designed to study a wide range of physics phenomena, including the Higgs boson and the top

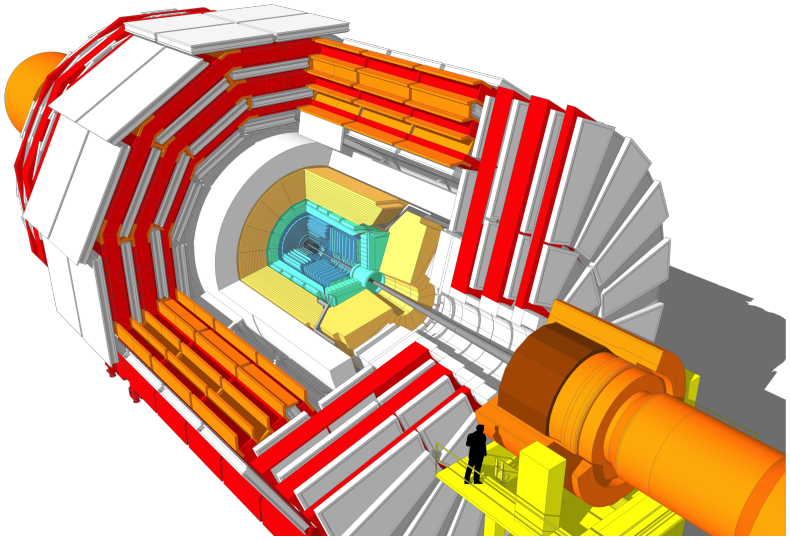


Figure 2.2: A 3D sketch of the CMS detector at the Large Hadron Collider. [37]

quark, and search for new particles beyond the Standard Model. The CMS detector is a cylindrical apparatus with a length of 21.6 m, a diameter of 15 m, and a weight of 14,000 tons. It is located 100 m underground at Point 5 on the LHC ring. Closest to the interaction point is the silicon tracker, which measures the trajectories of charged particles with high precision. The tracker is surrounded by electromagnetic and hadronic calorimeters, which measure the energy of electrons, photons, and hadrons. The calorimeters are followed by the superconducting solenoid magnet, which generates a magnetic field of 3.8 T. The magnet bends the trajectories of charged particles, allowing their momenta to be measured. The outermost layer of the CMS detector is the muon system, which identifies muons and measures their momenta. The CMS detector is designed to be hermetic, meaning that it covers the entire solid angle around the interaction point, and to do so most of its systems are split into a barrel section covering a 360° angle around the beam pipe and two endcaps that cover the front and back of the detector. This feature allows the detector to measure the total energy and momentum of particles produced in the collisions. A 3D sketch of the CMS detector is shown in Figure 2.2.

Trigger System

The CMS trigger system is a crucial component of the Compact Muon Solenoid (CMS) experiment at the Large Hadron Collider (LHC), responsible for efficiently selecting events of potential physics interest from the enormous number of proton-proton collisions. The system employs two levels of selection: the Level-1 (L1) trigger, implemented in custom hardware, and the High-Level Trigger (HLT), a software-based system running on a farm of commercial processors. [38] The L1 trigger rapidly analyzes coarse detector information to reduce the event rate from 40 MHz to about 100 kHz, while the HLT performs more sophisticated event reconstruction and selection,

further reducing the rate to approximately 2 kHz.

The L1 trigger identifies candidate objects, such as muons, electrons, photons, jets, and missing transverse energy, based on energy deposits and hit patterns in the calorimeter and muon detectors [38]. The HLT refines the identification and selection of these objects using algorithms similar to those used in offline analysis but with greater speed. The trigger system also includes a dedicated mechanism for identifying heavy stable charged particles (HSCP) relying on the timing characteristics of signals in the RPC muon detectors.

The performance of the trigger system is continuously monitored and optimized to adapt to changing LHC conditions and physics goals. The trigger menus, consisting of sets of selection criteria and algorithms, are adjusted to maintain high efficiency for signals of interest while keeping the trigger rates within the bandwidth limits of the readout electronics and data acquisition system. The CMS trigger system has been instrumental in the successful collection of data for a wide range of physics analyses, enabling the CMS experiment to achieve significant results, such as the observation of the Higgs boson and the measurement of the B0s $\rightarrow \mu^+\mu^-$ branching fraction.

Tracking System

The CMS tracking system measures the trajectories of charged particles generated in the proton-proton collisions. The magnetic field inside the detector causes the particles to bend their trajectories due to the Lorentz force, allowing the identification of the particle charge and momentum. The tracking system consists of multiple layers of silicon sensors. When a charged particle traverses such a silicon sensor, it will ionize the semi-conducting material, leading to the creation of electron-hole pairs, which results in a measurement current. The layers close to the center of the tracker form the pixel tracker, providing points in the three-dimensional space, thus enabling reconstruction of the primary vertices, i.e. the point of origin of the proton-proton collision, at high precision. It is composed of four layers in the barrel region and three discs of silicon sensors on each endcap. There are 124 million readout channels with a readout rate of 400 MB/s.

The pixel tracker employs a two-phase CO_2 cooling system to maintain optimal operational temperatures. Its design includes barrel layers positioned at radii of 29 mm, 68 mm, 109 mm, and 160 mm, along with three disks at each end located at distances of 291 mm, 396 mm, and 516 mm from the detector's center. The core structural unit is a silicon sensor module, which features 160×416 pixels, each measuring $100 \times 150\,\mu\text{m}^2$, with a pitch (the distance between the centers of two adjacent pixels) of 100 μm [39]. In total, the pixel tracker comprises 1856 such modules.

A crucial performance metric for tracking detectors, the spatial resolution, is influenced by factors such as the angle between the drift direction of electrons or holes in the tracking material, i.e. the Lorentz angle, and whether a particle interacts with multiple sensors in a single layer, which spreads the measured charge across adjacent cells. Under ideal conditions, where only one sensor per layer is impacted, the probability of a particle hitting a specific point within a sensor is uniformly distributed as

$p(x) = \frac{1}{d}$, where d represents the pitch. The expected value is calculated as:

$$\langle x \rangle = \int_{-d/2}^{d/2} x \cdot p(x)\, dx = \frac{1}{d} \int_{-d/2}^{d/2} x\, dx = 0 \tag{2.15}$$

The resolution, therefore, is derived as:

$$\sigma^2 = \langle (x - \langle x \rangle)^2 \rangle = \langle x^2 \rangle = \int_{-d/2}^{d/2} x^2 \cdot p(x)\, dx = \frac{d^2}{12} \tag{2.16}$$

$$\sigma = \frac{d}{\sqrt{12}} \tag{2.17}$$

In this configuration, the resolution of the pixel tracker is approximately $\sigma \approx 28.9\,\mu$m.

The silicon strip tracker (SST), forming the outer portion of the tracking system, comprises 9.3 million silicon strips distributed across 15,148 modules. The SST spans a length of 5 m and has a diameter of 2.5 m. In the barrel region, there are ten layers of strip modules, while the endcaps contain up to seven rings. The thickness of the silicon sensors varies, being 320 μm for the inner layers and 500 μm for the outer layers. The distance between the strips (pitch) ranges from 80 μm to 205 μm, resulting in spatial resolutions between 23.1 μm and 59.2 μm. The ratio of the pitch to the strip width is maintained at a constant value of 0.25 [40]. The barrel layer modules measure the r and ϕ coordinates, whereas the endcap layers measure ϕ and the longitudinal z coordinate.

The transverse momentum resolution of the tracking system for high-momentum tracks around 100 GeV is between 1% and 2% for a pseudorapidity of $|\eta| \approx 1.6$. The pseudorapidity is a spatial coordinate used to describe the angle of a particle relative to the beam axis and it is defined as:

$$\eta = -\ln[\tan\frac{\theta}{2}]. \tag{2.18}$$

Additionally, the transverse impact parameter resolution for high p_{T} tracks is 10 μm.

The CMS Calorimeters

A brief overview of the CMS Calorimeters now follows, but a more in-depth explanation of calorimeters and the technical aspects of calorimetry is given in Chapter 3. The CMS detector consists of two calorimeters: the electromagnetic calorimeter (ECAL), designed to measure the energy of photons and electrons, and the hadronic calorimeter (HCAL), which quantifies the energy of both charged and neutral hadrons. The ECAL is classified as a homogeneous calorimeter, where energy deposition and detection occur within the same active material [41]. This differs from a sampling calorimeter, which alternates between layers of high atomic mass material (to induce electromagnetic showers) and active media such as scintillators (to measure the resulting particle energy). Due to energy losses via ionization in the absorber layers, sampling calorimeters generally have lower energy resolution compared to homogeneous ones. The need

for precise energy measurements in the ECAL, particularly for studying Higgs bosons decaying into two photons, necessitated the choice of a homogeneous calorimeter over a sampling design.

In the CMS ECAL, the active medium consists of 61,200 lead tungstate ($PbWO_4$) crystals in the barrel region, with an additional 7,324 crystals in each of the endcaps. Avalanche photodiodes (APDs) are employed as photodetectors in the barrel, while vacuum phototriodes (VPTs) are used in the endcaps. Lead tungstate is particularly advantageous due to its high density of 8.28 g/cm^3 and short radiation length ($X_0 =$ 0.89 cm), defined as the mean distance over which an electron's energy is reduced by a factor of $1/e \approx 36.8\%$. These properties enable the construction of a compact, highly granular calorimeter. The length of the crystals is 230 mm, corresponding to 25.8 X_0. The cross-section area at the front face is 22x22 mm^2 and 26x26 mm^2 at the rear face. The front faces are at a distance of 1.29m from the detector center.
The energy resolution can be parametrized as $\left(\frac{\sigma}{E}\right)^2 = \left(\frac{S}{\sqrt{E}}\right)^2 \oplus \left(\frac{N}{E}\right)^2 \oplus (C)^2$, where $S = 2.8\%$, $N = 12\%$, and $C = 0.3\%$ are the stochastic, noise, and constant terms, respectively. The energy resolution is approximately 1% in the central region of the barrel for electrons with $E = 45$ GeV.

The CMS hadron calorimeter (HCAL) [42] serves to measure the energies of charged and neutral hadrons, as well as neutrinos and other exotic particles, ultimately contributing to the measurement of missing transverse energy, $E_{\mathrm{T}}^{\mathrm{miss}}$. Consequently, the HCAL plays a pivotal role in jet identification and energy measurement.

The innermost portion of the HCAL barrel is situated at a distance of 1.77 m from the detector center, extending outwards to 2.95 m. To ensure high-probability absorption of incident hadrons, an additional hadron calorimeter is positioned outside the solenoid to measure hadrons that have traversed the inner HCAL.

Unlike the CMS ECAL, the HCAL is a sampling calorimeter, employing alternating layers of brass absorber plates and plastic scintillators. The brass absorbers, composed of 70% Cu and 30% Zn, possess a density of 8.53 g/cm^3. This translates to a radiation length of 1.49 cm and a nuclear interaction length of 16.42 cm. The latter parameter signifies the mean distance traversed by a hadronic particle before undergoing an inelastic nuclear interaction. The HCAL energy resolution was measured in test beam studies for single pions and is found as [43]:

$$\frac{\sigma}{E} = \frac{52.9\%}{\sqrt{E}} + (5.7\%)^2. \tag{2.19}$$

The CMS Muon System

The CMS muon system is outside of the solenoid, built into the iron return yoke and is designed to identify and provide information to the trigger system on muons produced during collisions while also measuring their momentum. The strong magnetic field generated by the solenoid magnet enables precise momentum resolution. The system employs four different types of gaseous detectors for muon identification [44].

In the barrel region, drift tube (DT) chambers serve as detectors, arranged in four concentric cylindrical stations. The three innermost stations each house 60 chambers, while the outermost station contains 70 chambers. These chambers collectively include

approximately 172,000 wires, each about 2.4 m in length. A gas mixture of 85% argon and 15% carbon dioxide is used. To enhance detection efficiency, the drift cells within each chamber are arranged in an overlapping pattern, with each cell shifted by half its width relative to its neighbor. Each chamber comprises two or three "superlayers" (SLs), with each SL containing four layers of rectangular drift cells. In the outer two SLs, the wires are aligned parallel to the z direction, enabling measurements in the $r-\phi$ plane, whereas in the inner SLs, the wires are orthogonal to the beamline to measure the z coordinate.

In the endcaps, which detect muons with a pseudorapidity range of $0.9 < |\eta| < 2.4$, cathode strip chambers (CSCs) are employed. Each endcap contains four stations with a total of 468 CSCs. These chambers are arranged perpendicularly to the beamline to measure the $r-\phi$ plane. Based on multiwire proportional chamber technology, each CSC features six anode wire planes and seven cathode panels. The CSCs provide both muon measurement and triggering capabilities.

Resistive plate chambers (RPCs) complement the DTs and CSCs by improving background rejection and enabling precise beam crossing time measurements. Operating in avalanche mode, RPCs are gaseous parallel-plate detectors with excellent time resolution but coarser positional accuracy compared to DTs and CSCs. Notably, RPCs can tag ionizing events in significantly less time than the 25 ns between consecutive bunch crossings. In the barrel, six RPC layers are included—two in each of the first two stations and one in each of the last two. In the endcaps, one RPC layer is present in each of the first three stations.

To further enhance muon momentum resolution, a dedicated alignment system monitors the positions of the muon detectors relative to each other and the inner tracker. This design achieves a reconstruction efficiency of 95–99% across a broad angular range of $10^\circ < \theta < 170^\circ$. The momentum resolution for high-energy muons ranges from 1% to 8.25% in the barrel region and from 2% to 13.1% in the endcaps for transverse momenta (p_T) up to 2000 GeV, based on combined information from the tracker and muon systems. This resolution also depends on the alignment between the tracker and muon systems [45].

Thanks to its high efficiency and performance, the CMS muon system is an essential component of the detector. It facilitates the identification of processes involving muon decays, such as the $H \to ZZ^* \to 4\mu$ decay channel of the Higgs boson, which was pivotal in its discovery.

3 Calorimetry

Calorimeters are fundamental tools in modern particle physics. They are designed to measure the energy of a particle that passes through them by ideally fully absorbing their energy, therby destroying them. The general idea is that calorimeters are made of high-density materials, thus increasing the probability of particle interaction. While the particles lose their energy in the detector, they deposit charge or emit light proportionally to the amount of released energy. This proportion can be established via calibration.
Depending on the type of particles that are being detected, calorimeters can be of different types, but there are two major classes of them: electromagnetic calorimeters and hadronic calorimeters. Electromagnetic calorimeters are designed to measure the energy of electrons and photons, while hadronic calorimeters are designed to measure the energy of hadrons. In this chapter, the fundamentals of calorimetry will be introduced starting from how electromagnetic particles interact with matter, describing the general idea of electromagnetic and hadronic calorimeters, and some of the algorithms and simulation methods that are most commonly used in the field of high energy physics. Finally, there will be an introduction to the CMS HGCal calorimeter that is being constructed for the High Luminosity LHC upgrade and is of great relevance to this thesis project.

3.1 Particles Interactions with Matter

Charged Particles

When a charged particle passes through matter, it can interact with the medium in different ways. In general, the interaction can be either a deflection of a particle in the form of an elastic scattering or a loss of energy by the particle. These processes happen many times as the particle goes through the medium and it's the cumulative effect of such processes that can be measured. In addition, energy loss can also occur via other means, like Cherenkov radiation, nuclear reactions, or bremsstrahlung. The correct quantum-mechanical description of energy loss of a charged particle traveling through matter was first given by Bethe, Bloch, and other authors [46] in terms of momentum transferred, i.e. $-dE/dx$ with the formula [47]:

$$-\frac{dE}{dx} = 2\pi N_{\mathrm{a}} r_{\mathrm{e}}^2 m_{\mathrm{e}} c^2 \rho \frac{Z}{A} \frac{z^2}{\beta^2} \left[\ln \left(\frac{2 m_{\mathrm{e}} \gamma^2 v^2 W_{\max}}{I^2} \right) - 2\beta^2 - \delta - 2\frac{C}{Z} \right], \tag{3.1}$$

where N_{a} is Avogadro's number, r_{e} is the classical electron radius, m_{e} is the electron mass, c is the speed of light, ρ, Z and A are the density, atomic number and atomic mass of the absorbing material. z is the charge of the incident particle, β is the velocity of the incident particle in units of the speed of light, γ is the Lorentz factor, and v is

the velocity of the particle. $W_{\max}$ is the maximum energy transfer that is produced in a single head-on collision. For an incident particle of mass M, the maximum energy transfer is given by [47]:

$$W_{\max} = \frac{2m_e c^2 \eta^2}{1 + 2s\sqrt{1+\eta^2} + s^2},$$

with $s = m_e/M$ and $\eta = \gamma\beta$. I is the mean excitation potential that is in practice very difficult to calculate quantitatively and is usually taken from experimental data [47]. Finally, δ is the density effect correction, and C is the shell correction. The first one arises from the fact that the electric field of a charged particle tends to polarize the atoms in its path effectively reducing the contribution to the energy loss of electrons lying further away. C accounts for the effects that arise when the speed of the incident particle is comparable or smaller than the one of the electron bound to the atom. At these velocities, the bound electron can't be considered stationary with respect to the incident particle and the formula needs to be corrected.

The Bethe-Bloch formula is only valid for particles with $\beta\gamma \gg 1$. For particles with $\beta\gamma \ll 1$ the formula breaks down because a number of complicated effects come into play, such as the binding of the electrons in the atom, which are not accounted for in the Bethe-Bloch formula. Figure 3.1 shows how the energy loss behaves for different combinations of particles and absorbing material as a function of their velocity. At non-relativistic energies, the energy loss is dominated by the $1/\beta^2$ term and decreases until a speed of $v \simeq 0.96c$ where a minimum is reached. Particles in this region are called minimum ionizing particles (MIPs). At higher energies, the energy loss increases logarithmically with the velocity of the particle. For velocities below the MIP region, the $-dE/dx$ curves are distinct and dependent on the particle type, a characteristic that is used in particle identification.

An important observable that can be derived from the Bethe-Bloch formula is the range of a particle in a material. The range is defined as the average distance a particle travels before it comes to rest and it is necessary to determine the sizes of detectors used in experiments or, to determine the thickness of the protective shielding around the experiments. The range of a particle can be calculated by integrating the energy loss over the path length of the particle:

$$R = \int_0^T \left(\frac{dE}{dx}\right)^{-1} dE. \tag{3.2}$$

However, it turns out that this derivation is not exactly accurate as it doesn't account for the zigzag path that arises from the scattering of the particle as it passes through the material. The range of a particle can be calculated more accurately by using an empirical formula [47]:

$$R(T_0) = R_0(T_{\min}) + \int_{T_{\min}}^{T_0} \left(\frac{dE}{dx}\right)^{-1} dE, \tag{3.3}$$

where $T_{\min}$ is the minimum kinetic energy for which Equation 3.1 holds and $R_0(T_{\min})$ is an empirically determined constant that takes into account the low energy range behavior. If one knows the dE/dx of a particle in a material, it is possible to calculate

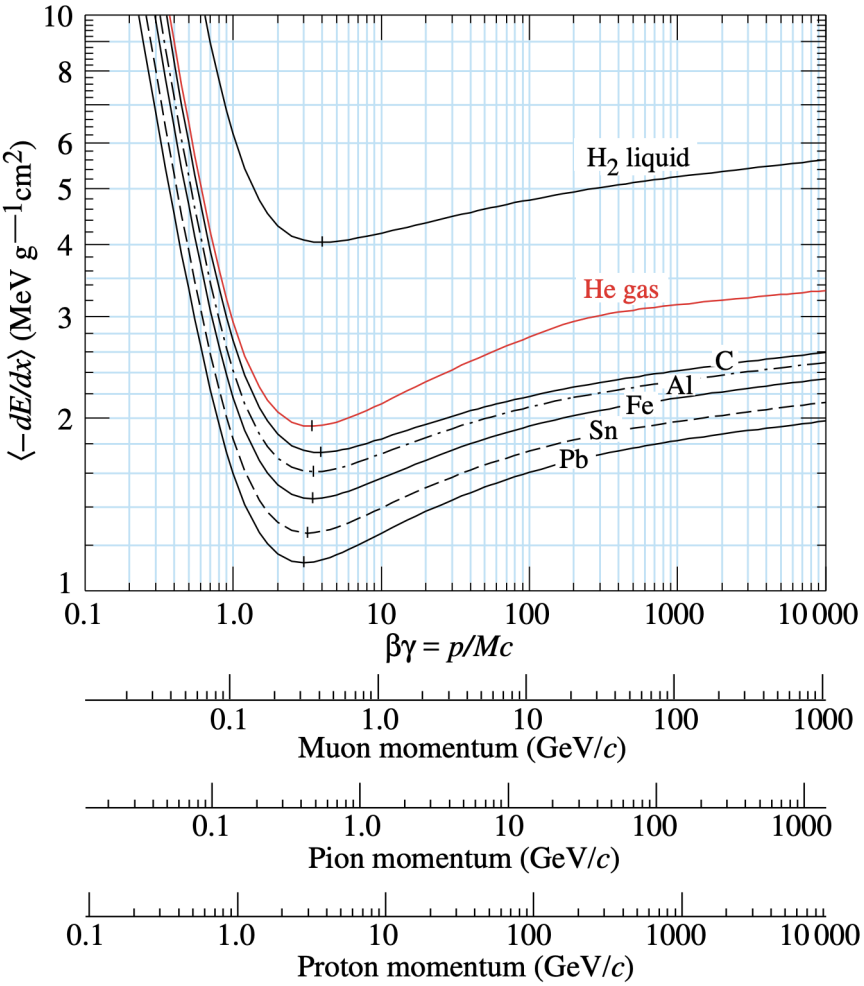


Figure 3.1: Energy loss of different particles as a function of their velocity in liquid hydrogen, gaseous helium, carbon, aluminum, iron, tin and lead. Taken from Ref. [48].

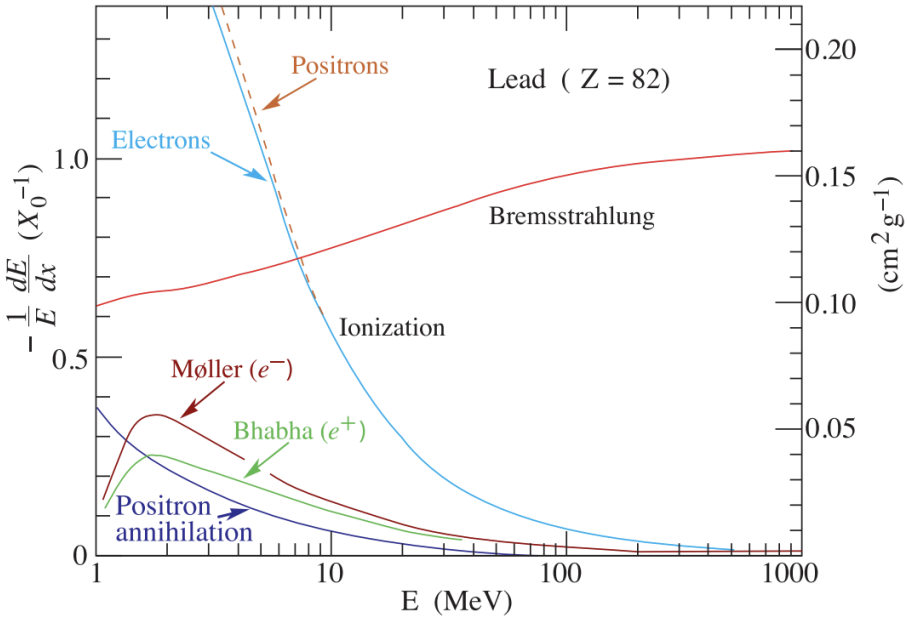


Figure 3.2: Fractional energy loss per radiation length in lead as a function of electron or positron energy. Electron (positron) scattering is considered as ionization when the energy loss per collision is below 0.255 MeV, and as Møller (Bhabha) scattering when it is above.. Taken from Ref. [48].

the energy loss of another particle in the same material by using the scaling law:

$$-\frac{dE_2}{dx}(T_2) = -\frac{z_2^2}{z_1^2}\frac{dE_1}{dx}\left(T_2\frac{M_1}{M_2}\right), \tag{3.4}$$

where T_2 is the kinetic energy of the second particle, z_2 and z_1 are the charges of the second and first particle, M_1 and M_2 are their masses. The range for different particles in the same medium can then be extrapolated with the scaling law:

$$R_2(T_2) = \frac{M_2}{M_1}\frac{z_1^2}{z_2^2}R_1\left(T_2\frac{M_1}{M_2}\right). \tag{3.5}$$

Electron/Positron interactions with matter

Electrons and positrons interact with matter in a number of ways. The most relevant ones are *ionization*, arising from the collision with atomic electrons and -as shown in Figure 3.2- more relevant at lower energies, and *bremsstrahlung*. Bremsstrahlung happens due to the scattering of the electron/position in the electric field of a nucleus. It can be interpreted as radiation arising from the acceleration of a charged particle when deflected by the electric field of the atomic nucleus. At lower energies, this process is a small factor in the energy loss, but it gains relevance with higher momentum, as Figure 3.2 points out. Since ionization is more relevant at lower energy regimes and bremsstrahlung at higher ones, it is possible to identify an energy value where the two contributions are equal. The *critical energy* (E_c) is defined as the energy of the electron for which is true that:

$$\left(\frac{dE}{dx}\right)_{ion} = \left(\frac{dE}{dX}\right)_{bremss},$$

and an approximate estimation for E_c in different materials is given by the empirical formula [49]:

$$E_c = \frac{800\ MeV}{Z + 1.2}.$$

There are other contributions to the energy loss like Møller and Bhabha scattering for the electron and the positron respectively. These refer to the scattering of the atomic electrons with energy loss below 0.255 MeV. Finally, electron-position annihilation $e^+e^- \rightarrow \gamma\gamma$ can occur when matter and anti-matter particles are transformed in two photons.

Photon interactions with matter

Photons interact with matter in three main ways: photoelectric effect, the Compton scattering, and pair production. The *photoelectric effect* is the process where a photon is absorbed by an atom and an electron is ejected from the atom. *Compton scattering* is the process where a photon scatters off an electron and loses energy while *pair production* arises when a photon creates an electron-positron pair. The probability of each of these happening depends on the energy of the photon and the material that the photon is passing through. The cross-section as a function of the energy for each of these processes is shown in Figure 3.3.

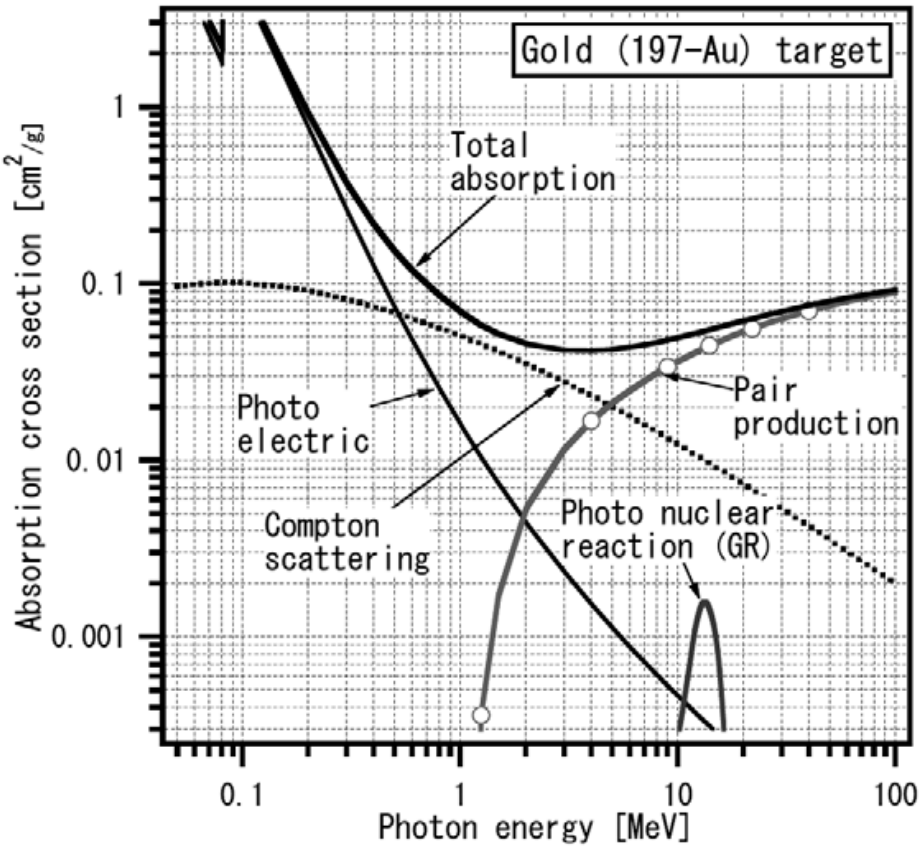


Figure 3.3: Cross sections for the photoelectric effect, Compton scattering, and pair production as a function of the energy of the photon in a gold target. Taken from Ref. [50].

3.2 Particle Showers

Electromagnetic showers

As just described, electrons and photons lose energy while passing through matter. The combined processes of bremsstrahlung for electron/positrons and pair production for radiation result into a cascade of particles that we call an *electromagnetic shower* and that is schematically represented in Figure 3.4. Such an electromagnetic shower is a complex process that is characterized by the production of a large number of particles in a short amount of time. The shower is initiated by the primary electron or photon and is composed of secondary electrons, positrons, and photons created through repeated bremsstrahlung and pair production. The entire process is stochastic and the number of particles produced in the shower is not deterministic. There are some parameters through which a shower can be characterized. For example, the radiation length is the average distance at which a high-energy electron loses all but $1/e$ of its energy by bremsstrahlung. The radiation length is a material-dependent quantity and can be calculated using the formula:

$$X_0 = \frac{716.4A}{Z(Z+1)\ln\left(287/\sqrt{Z}\right)}\ \mathrm{g/cm^{-2}}, \tag{3.6}$$

where Z is the atomic number of the material and A is the atomic mass. The radiation length is a useful quantity as it is used to determine the thickness of the calorimeter that is needed to contain the shower using an empirical measure to characterize the transverse shower development, which is the Molière radius ρ_M, i.e., the radius of a cylinder that contains 90% of the energy of the shower. The Molière radius is described as:

$$\rho_M = \frac{21\ \mathrm{MeV}}{E_c} X_0, \tag{3.7}$$

where E_c is the critical energy and $\mathrm{X_0}$ is the radiation length. Typically, these radiuses are of a few centimeters for different materials, for instance, 1.6 cm in lead, 4.3 cm in lead glass, and 9.1 cm in scintillator [49].

It is also possible to determine the energy resolution of the calorimeter which can be derived with the formula:

$$\frac{\sigma}{E} = \frac{a}{\sqrt{E}} \oplus \frac{b}{E} \oplus c, \tag{3.8}$$

where a is the stochastic term due to the fluctuations in the shower development, b is the noise term that takes into consideration the electronic noise of the detector, and c is the constant term making up for the non-uniformity of the detector. With X_0 and taking into consideration the asymptotic cross-section for photon interactions defined as:

$$\sigma(E \to \infty) = \frac{7}{9}\frac{A}{N_{\mathrm{A}} X_0}, \tag{3.9}$$

it is possible to derive another useful quantity, the *mean free path* λ_γ which is the average distance that a photon travels before interacting with the material. The mean free path is given by the formula:

$$\lambda_\gamma = \frac{9}{7} X_0. \tag{3.10}$$

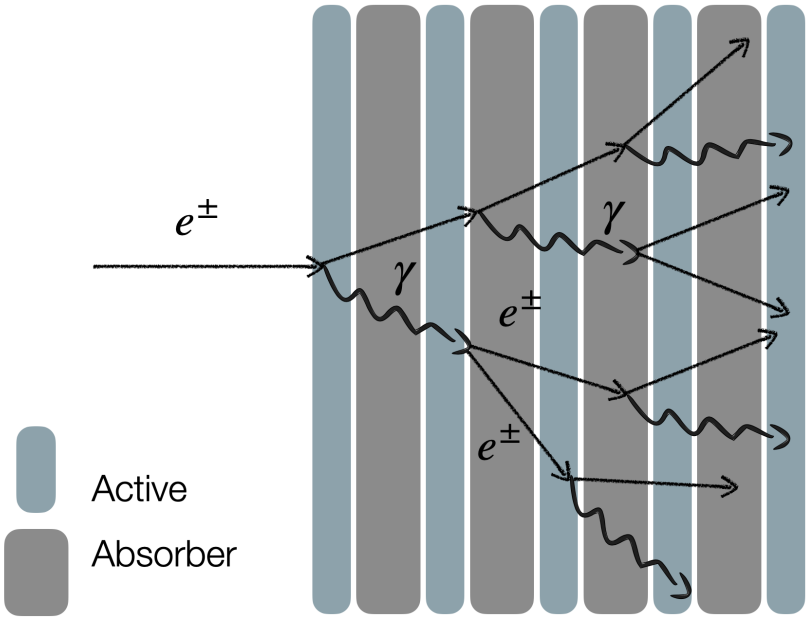


Figure 3.4: Schematic representation of an electromagnetic shower in a sampling calorimeter, which is described in Section 3.3. The shower is initiated by a high-energy electron or photon and is composed of secondary electrons, positrons, and photons.

Hadronic showers

Interactions involving hadrons can be categorized into hadronic and electromagnetic interactions. For charged hadrons, electromagnetic interactions resemble those of other heavy charged particles, as previously described. Neutral hadrons initially do not participate in electromagnetic processes. However, their hadronic interactions often generate charged hadrons, which are then able to interact electromagnetically. Hadronic interactions occur between energetic hadrons and the nuclei of a material, taking various forms. One such form is *spallation*, where a hadron undergoes inelastic scattering with protons and neutrons in the nucleus. Depending on the energy transferred, parts of the nucleus may also be ejected. This interaction leaves the nucleus in an excited state, which may lead to fission -where the nucleus splits, releasing additional energy— or to nuclear evaporation, where the nucleus releases excess energy in the form of α, β or γ radiation. The pions and ejected fragments generated through spallation can themselves carry enough energy to trigger further hadronic interactions, producing additional particles. This process repeats until the energy of the resulting particles is too low to form new pions, creating what is known as a hadronic shower. At this stage, any remaining particle energy that cannot contribute to pion production is dissipated through radiation or by capture into a nucleus. Unlike the relatively predictable outcomes of Bremsstrahlung or pair production, nuclear interactions produce a wide variety of particles, resulting in more complex hadronic shower patterns compared to electromagnetic showers. Additionally, the smaller number of interactions that lead to secondary particles in electromagnetic showers causes statistical fluctuations to have a more pronounced effect. On top of that, a hadronic shower can have an electromagnetic component as well, as the pions produced in the hadronic interactions can decay

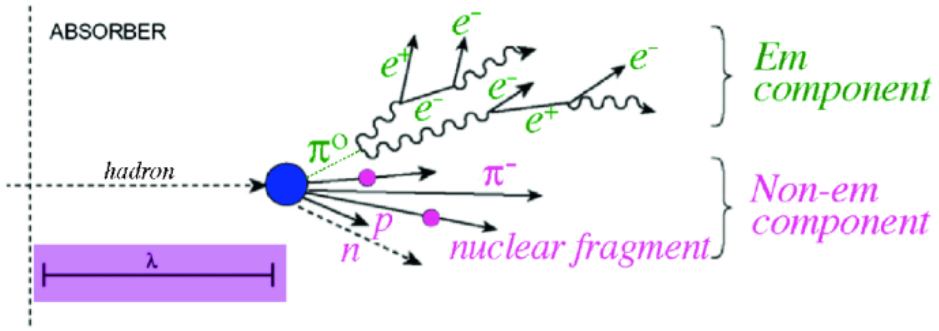


Figure 3.5: Schematic representation of a hadronic shower that carries both hadronic and electromagnetic components. Taken from Ref. [51].

into photons and electrons, as illustrated in Figure 3.5. This is why hadronic showers are more complex to reconstruct than electromagnetic showers. The length scale of a hadronic shower is given by the nuclear interaction length λ_{i} which is the average distance that a hadron travels before undergoing an inelastic collision and is generally defined as [49]:

$$\lambda_{\mathrm{i}} = \frac{1}{n\sigma_{had}}, \tag{3.11}$$

where σ_{had} is the cross-section for the hadronic interaction and n is the number of atoms per unit volume in the target material. The nuclear interaction length can be much larger than the radiation length, leading to larger showers that require bigger detectors to be contained.

3.3 Calorimeters

As introduced, calorimeters are physicists' instruments to measure particle energies. They do so by completely absorbing the particle, taking advantage of the well-known interactions with matter described in Section 3.1. The energy deposited into the calorimeter either as charge or as emitted photons can then be measured to perform a reconstruction of the particle energy. We saw that calorimeters can be divided into two main categories based on the particles that need to be measured.

In electromagnetic calorimeters, photon, and electron energies are primarily determined through electromagnetic showering, followed by ionization within the scintillating material. A high-energy photon interacts with the high-Z nuclei in the medium, producing an electron-positron pair. These particles are deflected by the nuclei's electric fields, emitting high-energy photons via bremsstrahlung. This process repeats, generating further electron-positron pairs, until the particles' energy drops below the threshold for pair production. At this stage, they deposit their remaining energy through ionization. In scintillating materials, ionization excites electrons to higher energy states, and their subsequent de-excitation emits photons, typically in the visible spectrum. Photodetectors capture this light, enabling an indirect measurement of the particle's energy.

On the other hand, hadronic showers exhibit significantly greater complexity than electromagnetic showers due to the presence of additional nuclear interactions. Key interaction processes include hadronization, nuclear spallation, and nuclear de-excitation,

which lead to the evaporation of soft nucleons. A considerable "invisible fraction" of energy arises from interactions, where energy is absorbed as nuclear binding energy or through recoil within the material. Furthermore, the production of neutral pions (π^0), which decay exclusively into photons and subsequently initiate electromagnetic showers, contributes to the energy deposition. These pions effectively deposit their entire energy via electromagnetic processes. Determining the fraction of electromagnetically deposited energy, f_{EM}, and disentangling the contributions from various nuclear interactions presents a significant challenge, requiring intricate Monte Carlo simulations for accurate calibration.

However, calorimeters can also be divided into two design categories: sampling and homogeneous calorimeters. In homogeneous calorimeters, the entire volume of the detector is built with a single material that is used to both absorb the particle and to measure the energy. This has the advantage that all the absorbed energy can be measured, but has several disadvantages such as the size of the calorimeter which then translates to a more difficult design and a higher cost. On the other hand, sampling calorimeters are built with alternating layers of absorber and active material. The absorber is usually a high-density material that is used to absorb the particle and it is alternated with an active material. This allows to reduce the size and cost of the calorimeter, at the disadvantage that only a fraction of the energy is absorbed and measured. Because most of the energy is dissipated in the absorber material, it is necessary to infer the energy of the particle using the fraction deposited in the active material. The fraction between the energy of the incident particle E_{full} and the energy deposited in the active material E_{active} is called the sampling fraction f_{sampling} and is defined as:

$$f_{\mathrm{sampling}} = \frac{E_{\mathrm{active}}}{E_{\mathrm{full}}}. \tag{3.12}$$

A low sampling fraction means uncertainty in the energy measurement which, as mentioned, is the main downside of such calorimeter designs. However, in most cases, it is necessary to find a balance between this and the cost-size factor of the detector. Another advantage of sampling calorimeters is that the division into layers allows the independent readouts of the layers which can be used to perform a longitudinal shower profile reconstruction, something that is not possible with homogeneous ones.

Another concept in calorimeters is *granularity*. Each layer is divided into cells that are used as independent readout channels, to produce a better "picture" of the shower. The size of the cells is called granularity and it determines the resolution of the calorimeter. A high granularity allows us to better reconstruct the shower radial profile. This is another parameter to tune in the design of a calorimeter: a bigger cell size means less detailed showers, but a lower cost and vice versa.

Since measuring the energy of a particle is the main goal of a calorimeter, one of the most important parameters is the energy resolution defined in Equation 3.2 and that gives a measure of the uncertainty in the energy measurement.

3.4 The CMS High Granularity Calorimeter

The CMS High Granularity Calorimeter (HGCal) [2] is a sampling calorimeter that is being designed for the High Luminosity LHC upgrade [1] to replace the existing endcap

calorimeters of the CMS detector. The HGCal, shown in Figure 3.6, is composed of two sub-detectors: the electromagnetic calorimeter (CEE) and the hadronic calorimeter (CEH) and it is designed to cover a pseudorapidity range from 1.5 to 3. It is designed to have a high granularity in both the longitudinal direction and in the layer planes. On top of that, the HGCal system will be able to discriminate the time of hits with a time resolution of about 25 ps for an energy deposit equivalent to a charge of 50 fC, as compared to a spread of time from collisions in a single bunch crossing of a couple of hundred ps [2].

Detector design and mechanics

All the HGCal layers have been designed with multiple 60° wedges called *cassettes* that are assembled together to form a full layer. In the CEE, the cassettes are made of silicon sensor modules placed on each side of a 6 mm thick copper cooling plate with onboard electronics (see Figure 3.7). The active material comprises hexagonal silicon sensors sandwiched into 1.4 mm thick WCu (75%,25%) plates that are used as absorbers. The silicon sensors have different sensitive thicknesses of 300, 200, and 120 μm in regions of increasing radiation levels. On top of that, silicon sensor cells have different sizes of $\sim$0.5, $\sim$1.2 cm^2, with the coarser region being further apart from the beam pipe. The CEE total thickness is $\sim$34 cm, i.e 26 X_0 and 1.3λ_{I}, and it is divided into 28 sampling layers. The CEH has 22 sampling layers divided into two parts: the front hadronic section with silicon as the active medium in the high radiation region and the back hadronic section with plastic scintillator as the active medium in the low radiation region. The active layers are sandwiched between layers of steel absorbers 35 mm thick in the front part and 68 mm thick in the back part. The CEH is $\sim$1.5 m long corresponding to a total depth of 8.7 λ_{I}. In Figure 3.8 we can see the different layouts of the CEE and CEH layers.

Precision Timing

Timing measurement for the Time-of-Arrival (ToA) is a great tool added for event reconstruction during the HL-LHC. The HGCal will be able to measure the time with a resolution of about 25 ps the time of electromagnetic and hadronic showers. Enforcing the compatibility between the ToA will help reject pile-up and identify primary vertices. Results from beam tests have shown that the time resolution is not subject to strong variability for different cell thicknesses when the resolution is measured as a function of the signal-to-noise ratio (S/N) nor does it vary as a result of irradiation up to fluences expected after 3000 fb^{-1} [2]. So the resolution can be expressed as:

$$\sigma_t = \sigma_{jitter} \oplus \sigma_{floor}, \tag{3.13}$$

$$\text{where } \sigma_{jitter} = \frac{A}{(S/N)}. \tag{3.14}$$

σ_{floor} is a constant term (a precision floor), and the symbol $\oplus$ denotes the quadratic summation. The constant A is fixed by the response time and noise characteristics of the sensor and a preamplifier.

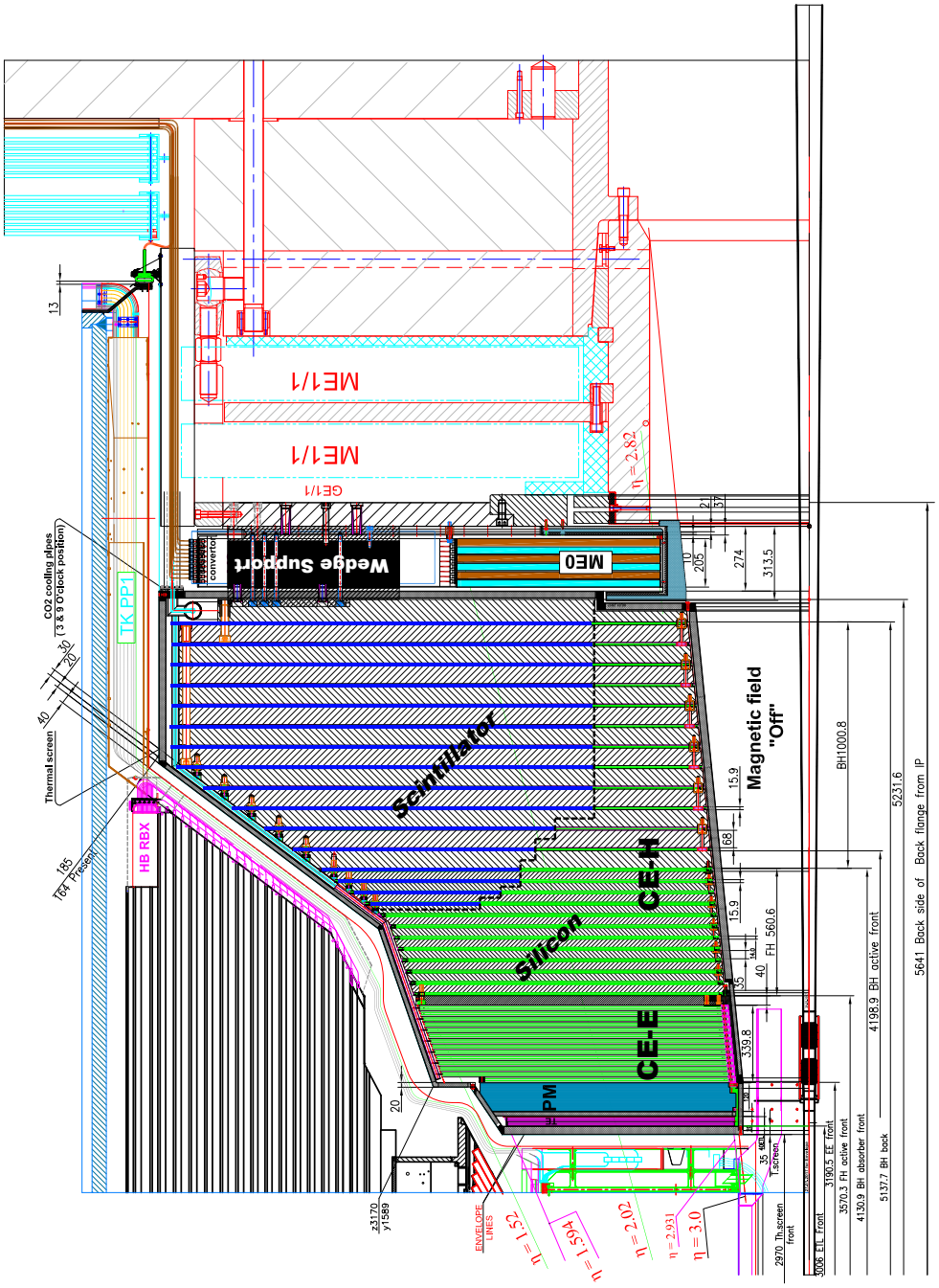


Figure 3.6: Schematic view of the HGCal from Ref. [2]. The green layers represent the silicon that acts as the sensitive part of the calorimeter. In the CEH, the scintillator (in blue) starts from the 9th layer and takes increasingly bigger portions of the hadronic calorimeter layer moving deeper into it.

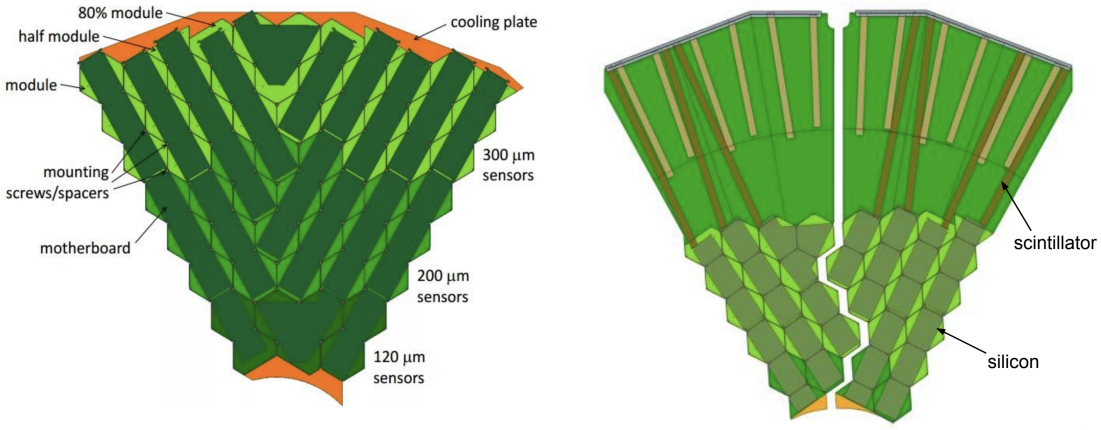


Figure 3.7: Schematic view of the HGCal cassettes from Ref. [52].

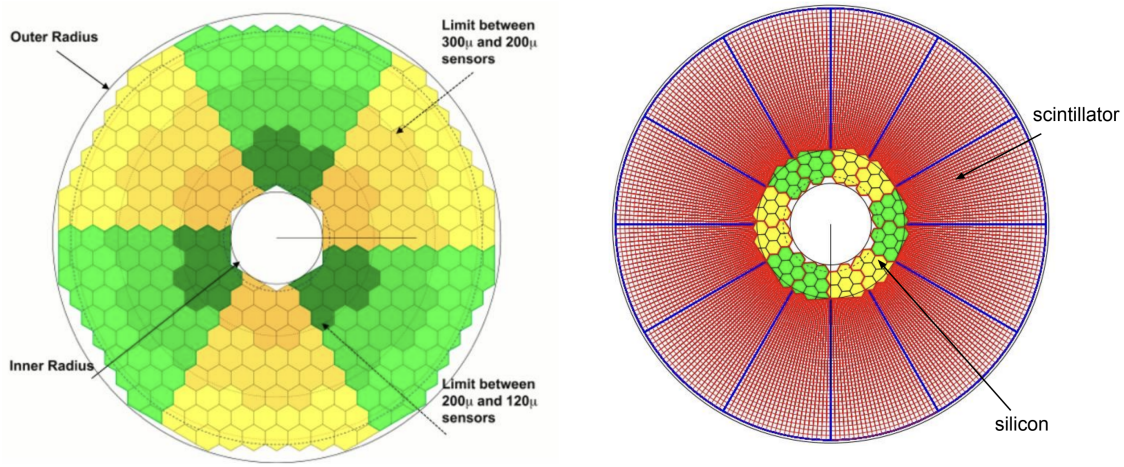


Figure 3.8: The 9^{th} CEE layer (on the left) is made solely out of silicon sensors and the 12^{th} CEH layer (on the right) is made out of scintillator and silicon sensors. Also different shades of yellow and green indicate the different thicknesses of the sensors. Taken from Ref. [52].

The topology and kinematics of the showers in the HGCal are the main factors that determine the effective timing performance. Electromagnetic showers have small containment radii and fast development and usually result in many cells with significant energy deposits. On the other hand, hadronic showers have a relatively wide core and tracks that develop laterally with respect to the shower axis. To make a time measurement it is required that at least three cells within a selection radius, ρ, of the shower axis, each with an energy deposit of >12 fC fire the ToA [2]. Since the ToA measurement is only possible with cells with deposited charge above 12 fC, it is subject to large event-to-event fluctuations. The time distribution of the energy deposited has long tails of later time measurements.

Pileup mitigation using timing information

In the Technical Design Report [2], it is shown how the inclusion of timing information can help with pileup mitigation. Figure 3.9 displays an example of hits of a simulated VBF event (qHH; H $\rightarrow\gamma\gamma$) with and without a timing requirement of $|\Delta t| > 90$ ps. The timing information clears up the projected hits allowing to clearly distinguish the jets, helping the development of jet reconstruction, and making the energy estimation less affected by pileup.

3.5 Calorimeter simulation

Simulating particle interactions within calorimeters is the most computationally demanding aspect of detector simulations. Traditional simulation tools use physics-based Monte Carlo (MC) methods to track each particle within the calorimeter medium and model all possible interactions. However, given the extensive simulation requirements of future collider experiments, relying solely on such detailed simulations will become unsustainable. To address this, various fast simulation methods have been developed to optimize or bypass parts of the simulation process. The generative machine learning models introduced in this thesis exemplify these fast simulation techniques, though several non-machine-learning approaches also exist. The following sections explore both full MC-based simulations and their faster alternatives.

Full Monte Carlo simulation

GEANT4 (GEometry ANd Tracking) [53–55], is the primary toolkit used in high energy physics to produce full Monte Carlo simulations of detector response. The simulation starts with a description of the geometry of the detector in use. This is done by adding a series of volumes and the material that they are made of. The geometry is then used to simulate the passage of particles through the detector. The simulation is done in steps during which the particle is propagated through the detector, and the interactions with the material are simulated. At each of these steps the software picks one of the possible interactions based on their probability. The probability of an interaction happening is characterized by the mean free path as follows:

$$p(l) = e^{\int_0^l \frac{1}{\lambda(l)} dl}, \tag{3.15}$$

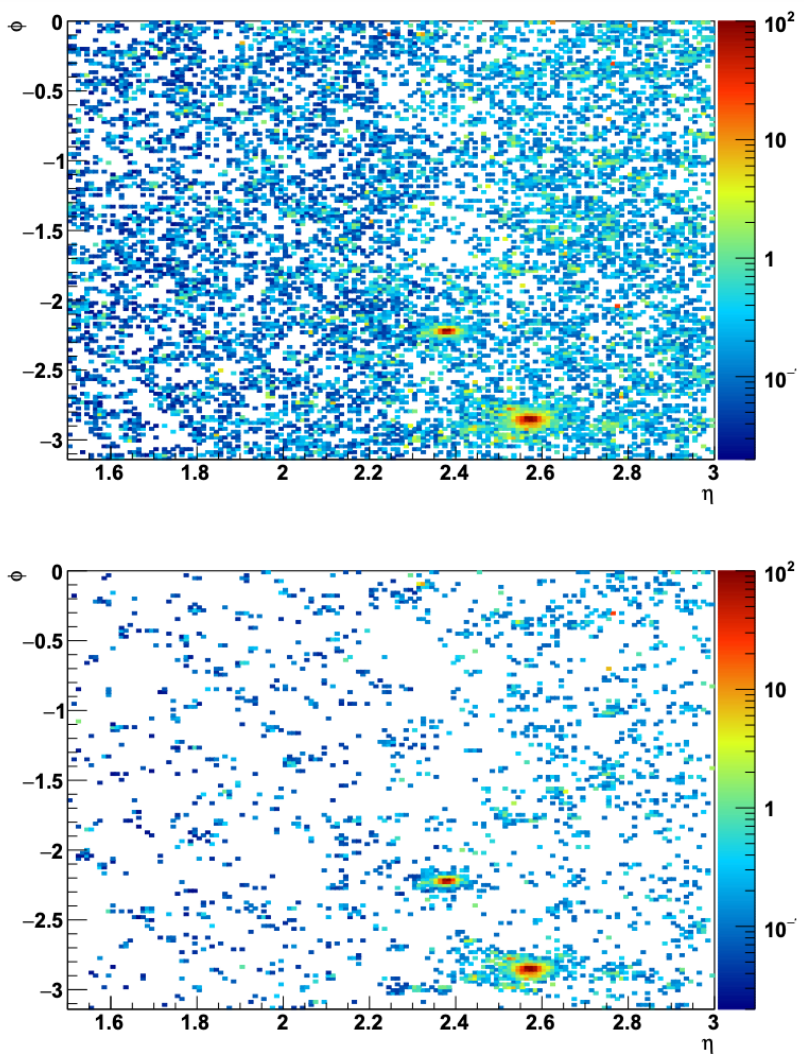


Figure 3.9: VBF($H \to \gamma\gamma$) event with one γ and VBF jet in the same quadrant (shown as red spots). The comparison of the two plots shows the effect of the timing information on the pileup mitigation. The upper plot shows the hits projected on the front face of the calorimeter with a charge larger than the 12 fC threshold and without any timing requirements, while in the lower plot, a timing requirement of $|\Delta t| > 90$ ps has been imposed. Taken from Ref. [2].

where $\lambda(l)$ is the mean free path of the particle in the material and l is the distance traveled. The parameter $\lambda(l)$ is described for most of the interactions by:

$$\frac{1}{\lambda} = \rho \sum_i \frac{x_i \sigma_i}{m_i} \tag{3.16}$$

where ρ is the density of the material, x_i is the fraction of the material that is made of the i^{th} element, σ_i is the cross-section of the interaction and m_i is the atomic mass of the i^{th} element. To perform a random sample of the path length of the process, it is possible to rewrite Equation 3.5 as:

$$\begin{aligned} p(l) &= e^{-n_\lambda}, \text{ with} \\ n_\lambda &= \int_{x_0}^{x_1} \frac{1}{\lambda(x)} dx, \end{aligned} \tag{3.17}$$

allowing for sampling from the exponential distribution via the sampling of η from a uniform distribution between 0 and 1 and setting:

$$n_\lambda = -\ln \eta. \tag{3.18}$$

from which the *total step size* can be calculated as:

$$s(x) = n_\lambda \cdot \lambda(x). \tag{3.19}$$

the total step size s is then calculated for all possible processes and the shortest step is selected as the next step of the particle. The process is repeated until the particle reaches the end of the detector or until the particle is absorbed. GEANT4 is very accurate as it tracks each particle individually, but this makes it very computationally expensive. The processes that are simulated come from a *physics list* which defines processes and cross-sections for every specific particle.

Fast Simulation

Fast Simulation (FastSim) approaches are used to tackle the challenge of the simulation speed. Some of these approaches are described here. Delphes [56] is a software framework designed for the rapid simulation of entire detectors. Instead of modeling individual particle showers, Delphes assigns the energy of particles reaching the calorimeter to a single ECAL and a single HCAL cell at the particle's impact position. The distribution of energy between the ECAL and the HCAL depends on the particle type. For instance, electrons and photons deposit all their energy in the ECAL, while stable hadrons deposit theirs entirely in the HCAL. To emulate the imperfect resolution of real calorimeters, the deposited energy is smeared using a factor derived from the calorimeter's resolution. This approach enables extremely fast simulations. However, Delphes requires prior knowledge of the calorimeter's energy resolution, making it unsuitable for first-principles simulations. Furthermore, it only models the total energy, without accounting for shower shapes, which limits its compatibility with particle flow techniques.

GFLASH [57, 58] is a parameterized shower simulation method built on GEANT4. It relies on longitudinal and transverse energy profiles obtained from fully simulated

showers in GEANT4. Using Monte Carlo techniques, GFLASH places energy depositions in the simulated calorimeter to accurately reproduce these profiles and preserve their internal correlations. This method is considerably faster than performing a full simulation for every particle, but they are less accurate.

Frozen Showers [59] involves pre-simulating a library of showers generated by low-energy particles using a full simulation. During the main simulation, highly energetic particles are handled using either a full simulation or a parameterized method. Once a particle's energy falls below a predefined threshold, its simulation is terminated, and a pre-generated shower with matching energy, angle, and position is placed at its location. This method bypasses the need to simulate the numerous low-energy particles that constitute most of a shower, significantly reducing computational effort. A similar concept was earlier implemented by the H1 collaboration [60], which utilized a library of pre-simulated showers for fast simulation of entire showers instead of sub-showers.

Finally, machine learning techniques are been under constant development in recent years as they might prove to be a faster, cheaper, and more accurate solution to the problem of fast simulation. The next section will introduce some of the most common techniques used in the field of machine learning in general and applied to high-energy physics.

4 Machine Learning

With the term *Machine Learning* (ML) we refer to a set of algorithms that can learn from data as opposed to being explicitly programmed to perform a task. ML has been studied for many years now, but its need for high computational power and large datasets has only been met in recent years, with the advent of graphics processing units (GPUs) and big data.
ML algorithms can be divided into three main categories: supervised, unsupervised, and reinforcement learning. In supervised learning, the algorithm is trained on a labeled dataset, where each example is associated with a label. An example could be a set of images of animals and their corresponding labels (e.g., cat, dog, horse). In this case, the algorithm, after being trained on this labeled dataset, predicts the label of an image that has never been seen before. The most common supervised learning tasks are classification (as in the example above), and regression where the model tries to predict a value given an input.
In unsupervised learning, the algorithm is trained on an unlabeled dataset to learn the underlying structure of the data. Common examples are anomaly detection and generative models, where the algorithm tries to generate new data that is similar to the training data. This will also be the focus of this thesis.
Finally, reinforcement learning is a type of ML where an agent learns to interact with an environment by performing actions and receiving rewards. A common example is a model that learns how to play a video game by maximizing the score.
In this chapter, we will introduce the basic concepts of ML, underlying methodology, and evaluation metrics.

4.1 Gradient Descent

The concept of ML is based on the idea of defining a model $f(x)$, with $f : \mathbb{R}^n \to \mathbb{R}^m$, that is iteratively updated to describe the set of data points $X = \{x^1, x^2, ..., x^N\} \in \mathbb{R}^n$. The model is just a non-linear function that maps the n-dimensional input space to the m-dimensional output space. The function is defined by multiple parameters ω, that are optimized from the data to improve the model and help it perform the required task. This optimization is done by minimizing a loss function $L(\omega)$, which measures the difference between the predicted output of the model and the task-specific target of the model. More details on this will follow in the next sections. For now we can think about the loss as a function of the parameters ω and the data. So we can write the loss function as $L(\omega, X)$, and to minimize it we would need to take the gradient of such a function with respect to the parameters ω. It would look like this:

$$\nabla_{\omega,i} L(\omega, X) = \frac{\partial L(\omega, X)}{\partial \omega_i} \tag{4.1}$$

where ∇ is the nabla operator that returns the gradient of L, and $\frac{\partial L(\omega,X)}{\partial \omega_i}$ is the partial derivative of the loss function with respect to the i-th parameter ω_i.

Note that the gradient of the loss function is a vector that points in the direction of the steepest ascent of the function. The most common optimization algorithm used in ML is based on gradient descent, which is built on the idea of iteratively updating the parameters in the opposite direction of the gradient. The update rule is defined as:

$$\omega^{t+1} = \omega_t - \alpha \nabla L(\omega_t, X) \tag{4.2}$$

where ω_t is the value of the parameters at iteration t, and α is the learning rate, which is a hyperparameter that controls the size of the step taken in the direction opposite of the gradient. The learning rate is a crucial hyperparameter that needs to be tuned carefully, as a too-small value can lead to slow convergence, while a too-large value can lead to divergence. The gradient descent algorithm is simple and effective, but it has some limitations, such as the fact that it can get stuck in local minima, and it can be slow to converge when the loss function is non-convex.

Stocastic Gradient Descent

To compute one step of the gradient descent algorithm, we need to compute the gradient of the loss function with respect to all the parameters which might prove to be computationally expensive. Stochastic Gradient Descent (SGD) [61] is an optimization algorithm that addresses this issue by computing the gradient of the loss function with respect to a set of disjoint subset of the data points $X_j = \{x_k\}_{k=b_{j-1}}^{b_j}$, called mini-batches. This allows for more complex models and larger datasets to be trained in a reasonable amount of time. On top of that, the mini-batches can be shuffled introducing some randomness in the optimization process, which can help to escape local minima.

4.2 Optimizers

To overcome the limitations of the simple gradient descent algorithm, several optimization algorithms have been proposed in the literature. These algorithms are based on the idea of adapting the learning rate during training or using more sophisticated update rules to speed up convergence.

Momentum

Both SGD and Gradient Descent can be slow to converge when the loss function has a complex landscape, with many local minima and saddle points. To tackle this problem, one of the most popular optimization algorithms is momentum, which is based on the idea of adding a momentum term to the update rule. The momentum term is a moving average of the gradients, and it helps speed up convergence and avoid local minima. The update rule for the momentum algorithm is defined as:

$$v_{t+1} = \beta v_t + (1-\beta) \nabla L(\omega_t, X) \tag{4.3}$$

$$\omega_{t+1} = \omega_t - \alpha v_{t+1} \tag{4.4}$$

where v_t is the momentum term at iteration t, and β is the momentum parameter that controls the weight of the moving average. The momentum algorithm is more robust than the simple gradient descent, and it can converge faster and more reliably to the global minimum.

Adam

Adam [62] is another popular optimization algorithm that combines the ideas of momentum and adaptive learning rates. Adam expands on the momentum algorithm by adding an adaptive learning rate for each parameter, which is based on the first and second moments of the gradients denoted by m_t and v_t respectively [61]. These moments are computed as:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L(\omega_t, X) \tag{4.5}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)[\nabla L(\omega_t, X)]^2 \tag{4.6}$$

where β_1 and β_2 are the exponential decay rates for the first and second moments, respectively. The suggested default values for these parameters are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initialization of the moments is done by setting $m_0 = 0$ and $v_0 = 0$. Depending on the implementation, the moments can be bias-corrected by dividing them by $1-\beta_1^t$ and $1-\beta_2^t$ respectively:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{4.7}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{4.8}$$

Finally, the update rule for the Adam algorithm is defined as:

$$\omega_{t+1} = \omega_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4.9}$$

where ϵ is a small constant that is added to the denominator to avoid division by zero. This means that the update direction is given by the running average of the previous updates. On top of that, the learning rate is scaled by the square root of the second moment, which helps to adapt the learning rate to the curvature of the loss function. This allows Adam to converge faster and more robustly than other optimization algorithms. Additionally, the adaptive scaling of the learning rate makes Adam less sensitive to the choice of the learning rate hyperparameter, enabling a more stable training process.

4.3 Loss Functions

The loss function is a crucial component of the ML model, as it measures the difference between the predicted output of the model and the task-specific target. The choice of the loss function depends on the task at hand, and it can have a significant impact on the performance of the model. In this section, we will introduce some common loss functions used in ML.

Mean Squared Error

The Mean Squared Error (MSE) is one of the most common loss functions used in regression or classification tasks. It measures the average squared difference between the predicted output of the model and the true output of the data. The MSE is defined as:

$$L(\omega, X) = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{4.10}$$

where $f(x_i)$ is the predicted output of the model for the i-th data point, y_i is the true output of the data, and n is the number of data points in the dataset. The MSE is a convex function, which means that it has a single global minimum, and it is differentiable, which makes it suitable for optimization with gradient-based algorithms.

Binary Cross-Entropy

The Binary Cross-Entropy (BCE) is a loss function employed in binary classification tasks. It evaluates the performance of a classification model by assessing the discrepancy between the predicted probabilities and the actual class labels, which are either 0 or 1. The BCE is defined as:

$$L(\omega, X) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(f(x_i)) + (1 - y_i)\log(1 - f(x_i)) \tag{4.11}$$

where $f(x_i)$ is the predicted probability of the model for the i-th data point class, y_i is the true label of the data, and n is the number of data points in the dataset.

4.4 Evaluation Metrics

The evaluation of ML models is a crucial step in the development process, as it allows us to measure the performance of the model and compare it with other models. In this section, we will introduce some common evaluation metrics used in ML.

Accuracy

Accuracy is one of the most common evaluation metrics used in classification tasks. It measures the proportion of correctly classified data points in the dataset. The accuracy is defined as:

$$\text{Accuracy} = \frac{\text{Number of correctly classified data points}}{\text{Total number of data points}} \tag{4.12}$$

The accuracy is a simple and intuitive metric, but it can be misleading in imbalanced datasets, where one class is much more frequent than the other. In this case, the accuracy can be high even if the model is not performing well in the minority class.

Precision and Recall

Precision and recall are two evaluation metrics used in binary classification tasks. The precision measures the proportion of correctly classified positive data points among all data points classified as positive. The precision is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.13}$$

The recall measures the proportion of correctly classified positive data points among all positive data points in the dataset. The recall is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.14}$$

Precision and recall are complementary metrics, and they are often used together to evaluate the performance of a model. The precision measures the ability of the model to avoid false positives, while the recall measures the ability of the model to avoid false negatives.

Wasserstein Distance

The Wasserstein distance [63], also known as the Earth Mover's Distance, is a metric used to measure the distance between two probability distributions. It's commonly used in generative models to measure the similarity between the generated samples and the true data distribution. The Wasserstein distance is defined as:

$$W(p,q) = \inf_{\gamma \in \Pi(p,q)} \int_{\mathbb{R}^n \times \mathbb{R}^n} \|x-y\| d\gamma(x,y) \tag{4.15}$$

where p and q are the two probability distributions, $\Pi(p,q)$ is the set of all joint distributions with marginals p and q, and γ is the joint distribution. The Wasserstein distance measures the minimum amount of work required to transform one distribution into the other.

Kullback-Leibler Divergence

Another metric commonly used in generative tasks is the Kullback-Leibler (KL) divergence. The KL divergence is a measure of how one probability distribution differs from a second, reference probability distribution and it is defined as:

$$D_{\text{KL}}(p||q) = \int p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \tag{4.16}$$

where p and q are the two probability distributions. The KL divergence measures the information lost when the distribution q is used to approximate the distribution p. It is not symmetric, meaning that $D_{\text{KL}}(p||q) \neq D_{\text{KL}}(q||p)$.

4.5 Machine Learning Challenges

We have seen some of the techniques for Machine Learning that have been successfully applied to a wide range of tasks, from image recognition to natural language processing. However, several challenges need to be addressed to make ML more effective and reliable. In this section, we will introduce some of the main challenges of ML.

Overfitting

Overfitting is a common problem in ML, where the model learns the noise in the training data instead of the underlying pattern. This leads to poor generalization performance, as the model performs well on the training data but poorly on unseen data. Overfitting can be caused by a model that is too complex for the data, or by a lack of regularization. There are several ways to prevent overfitting, such as using simpler models, adding regularization terms to the loss function, or using dropout.

Underfitting

Underfitting is the opposite of overfitting, where the model can not capture the underlying pattern in the data. This leads to poor performance on both the training and test data. This can be caused by a model that is too simple for the data, or by a lack of training data. There are several ways to prevent underfitting, such as using more complex models, increasing the size of the training data, or using more powerful optimization algorithms.

Vanishing and Exploding Gradients

Vanishing and exploding gradients are common problems in deep learning, where the gradients of the loss function with respect to the parameters become very small or very large. This can lead to slow convergence or divergence of the optimization algorithm. Vanishing gradients are caused by the use of activation functions with small gradients, such as the sigmoid or tanh functions. Exploding gradients are caused by the use of activation functions with large gradients, such as the ReLU [64] function. There are several ways to prevent vanishing and exploding gradients, such as using activation functions with bounded gradients, batch normalization [65], or gradient clipping.

4.6 Neural Networks

Neural Networks (NN) are a class of ML models inspired by the structure of the human brain. They are composed of layers of interconnected neurons, where each neuron computes a weighted sum of its inputs and applies an activation function to the result. The output of the neuron is then passed to the next layer of neurons until the final output is produced. In this section, we will introduce some common types of Neural Networks layers and architectures.

Dense Layer

The Dense layer is the most common type of layer used in Neural Networks. It is composed of a set of neurons, where each neuron computes a weighted sum of its inputs and applies an activation function to the result:

$$f(x) = \sigma(Wx + b) \tag{4.17}$$

where $f(x)$ is the output of the neuron, σ is the activation function, W is the weight matrix, x is the input vector, and b is the bias vector. The output of the neuron is then passed to the next layer of neurons until the final output is produced.

Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a class of Neural Networks that are designed to process structured data, such as images or videos. They are composed of layers of neurons that apply convolutional filters to the input data, followed by pooling layers that reduce the spatial dimensions of the data. Some key concepts of CNN include convolution which is a process involving applying a filter on small regions of the input data. This filter slides through the input creating a map of the features highlighting specific patterns or characteristics of them. Applying the filter to multiple locations of the input means that the model's weight is shared leading to another key feature of the CNNs, which is translation invariance, i.e. the model is insensitive to shifts or translation of patterns within the input. Another important concept is pooling, which is a process that reduces the spatial dimensions of the data by aggregating the values of neighboring pixels. This helps to reduce the computational complexity while maintaining important information. An example of this process can be seen in the figure 4.1. In this section, we will introduce some common types of layers used in CNNs.

Convolutional Layer

The Convolutional layer is the core building block of Convolutional Neural Networks. It applies a set of convolutional filters to the input data, which allows the model to learn spatial patterns in the data. The output of the Convolutional layer is a set of feature maps, where each feature map corresponds to a different filter. The layer can be defined as:

$$f(x) = \sigma\left(\sum_{i=0}^{N}(W * x_i) + b\right) \tag{4.18}$$

where $f(x)$ is the output of the Convolutional layer, σ is the activation function, W is the convolutional filter, x_i is the i-th subset of the input data, N is the total number of subsets, and b is the bias vector.

Pooling Layer

The Pooling layer is a type of layer used in Convolutional Neural Networks to reduce the spatial dimensions of the data. It applies a pooling operation to the input data,

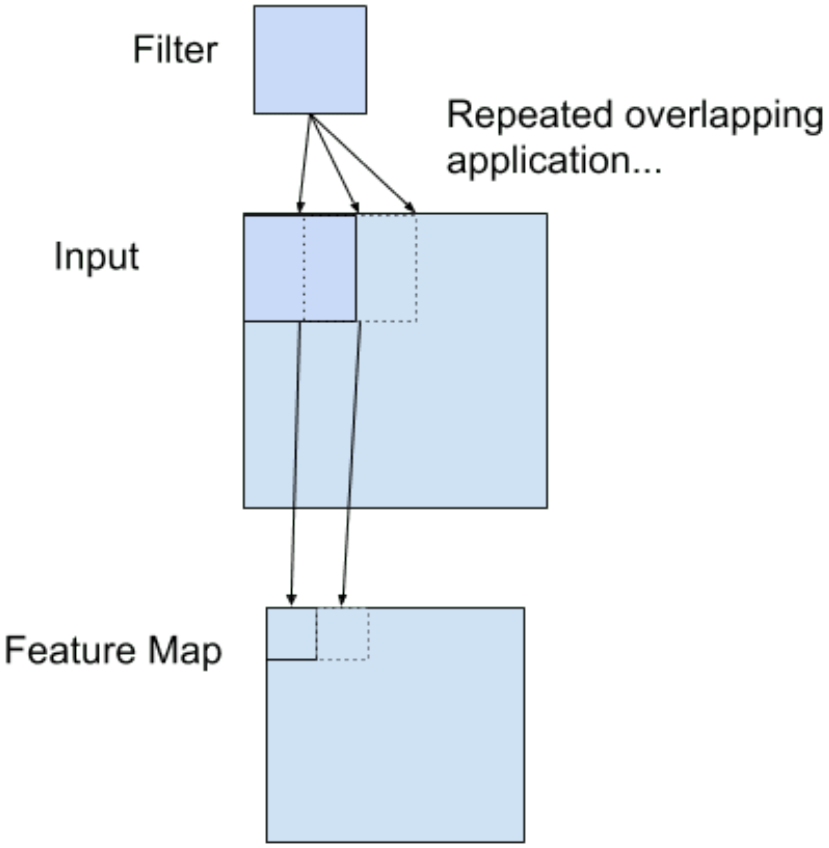


Figure 4.1: Example of a filter applied to a two-dimensional input to create a feature map. The filter slides through the input data, creating a map of the features. The image is taken from [66]

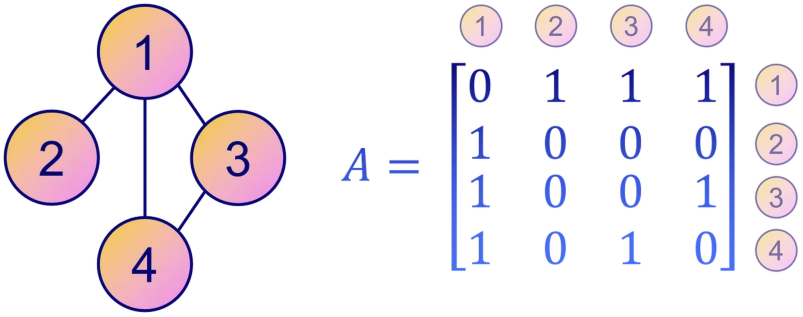


Figure 4.2: Example of an adjacency matrix of a graph. The matrix is symmetric, with 1s indicating the presence of an edge between two nodes. The image is taken from [67]

which aggregates the values of neighboring pixels. The most common pooling operation is the max pooling, which takes the maximum value of a set of pixels. Other examples are the sum pool or the mean pool that, as suggested by their name, takes respectively the sum and the average of the scanned pixels.

Graph and Point Cloud Neural Networks

Graph Neural Networks (GNN) are models that operate on structured data called graphs. A graph is defined as $G = (V, E)$ where V are the nodes and E are the edges connecting the nodes and giving a description of the relationship between them. $A \in \mathbb{R}^{NxN}$, with $N = |V|$ is the adjacency matrix of the graph, where $A_{ij} = 1$ if there is an edge between nodes i and j, and $A_{ij} = 0$ otherwise. Fig. 4.2 shows an example of an adjacency matrix.

There are different kinds of graphs depending on their structure [68]:

- **Directed/Undirected** Graphs: if the edges have a direction, i.e. they go from one node to another providing more information about the relationship between the nodes. The undirected scenario can be seen as a directed graph where the edges are bidirectional.
- **Homogeneous/Heterogeneous** Graphs: if the nodes are of the same type, the graph is homogeneous, otherwise it is heterogeneous.
- **Static/Dynamic** Graphs: Graphs can evolve over time, making the latter an important feature to keep track of in dynamic graphs.

These scenarios can be combined, meaning that a graph can be directed, static, and heterogeneous at the same time.

Design of the Loss Function

For GNN the target of the loss function can vary with the problem at hand. For graph classification, there are usually three types of tasks:

- **Node Classification**: the task focuses on nodes, categorizing them into classes, and performing regression or clustering.

- **Graph Classification**: the task focuses on the overall structure of the graphs.
- **Link Prediction**: in this case the focus is on the edges, classifying them or predicting whether there is an edge between two nodes.

Message Passing Layer

The Message Passing Layer is the core idea behind GNNs and all the variety of layers developed for this kind of network. The idea is to have a *permutational-equivariant* layer, meaning its output varies according to the variation of the input, that maps the features in an updated version of the graph, based on the information of the neighbors indicated by A. Let J_u be the neighbors of node $u \in V$, x_u the features of the node, and e_{uv} the features of edge (u, v). The message-passing layer can be defined as:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in J_u} \psi(\mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{uv})\right), \tag{4.19}$$

where ϕ and ψ are differentiable functions that can be identified as the neural networks, and $\bigoplus$ is a permutation-invariant function that aggregates the information from the neighbors, e.g. the sum, mean, or max of the elements.
Point Cloud networks are closely related to graph neural networks, as they operate on unstructured data, such as 3D shapes of objects, that are invariant over rotations and translations. The main idea behind Point Cloud networks is to treat the points as nodes of a graph, but no edges are defined and no adjacency matrix is used. The points are treated as unordered sets, and the network is trained to learn the underlying structure of the data.

4.7 Conclusion

In this chapter, we have introduced the basic concepts of Machine Learning, underlying methodology, and evaluation metrics. We have also discussed some common optimization algorithms used in ML, such as Gradient Descent, Momentum, and Adam. We have introduced some common loss functions used in ML, such as Mean Squared Error and Binary Cross-Entropy. We have also introduced some common evaluation metrics used in ML, such as Accuracy, Precision, and Recall. We have discussed some common challenges of ML, such as Overfitting, Underfitting, and Vanishing and Exploding Gradients. We have introduced some common types of Neural Networks layers and architectures, such as Dense Layers and Convolutional Neural Networks. Finally, we have introduced the basic concepts of Generative Adversarial Networks, Graph Neural Networks, and PointCloud Neural Networks. In the next chapter, Generative machine learning and some key concepts related to it will be introduced.

5 Generative Machine Learning

In recent years, generative machine learning has become an increasingly popular area of research in machine learning and artificial intelligence. Generative models are used to generate new data points that follow the same training data distribution and have a wide range of applications, including image generation, text generation, and data augmentation. Some popular examples of such models known to the public are OpenAI's DALL-E and GPT, NVIDIA's StyleGAN, and Stability AI's Stable Diffusion. These same concepts can be applied to high-energy physics to speed up the classical simulation processes based on Monte Carlo methods.

This chapter introduces the concept of generative machine learning (GML) and some of its applications, followed by some of the most popular generative models, including generative adversarial networks (GANs), variational autoencoders (VAEs), and diffusion and flow models.

5.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN)[69] are a class of Neural Networks designed to generate new data samples that follow the training data distribution. They are composed of two networks: a generator network that generates new data samples, and a discriminator network that discriminates between real and generated data samples. The generator network is trained to generate data samples indistinguishable from the training data. In contrast, the discriminator network tries to discern between real and generated data samples. GANs are powerful models that can learn complex patterns in the data, and they have been successfully applied to a wide range of tasks, from image generation to text generation. In this section, the basic concepts of GANs will be introduced.

Generator and Discriminator Networks

The Generator (G) network is a Neural Network that takes a random noise vector as input and generates new data samples and can be defined as $G(z,\theta_G)$, where z is the random noise vector and θ_G are the parameters of the Generator network. The Discriminator (D) network takes a sample as input and discriminates between real and generated samples and can be seen as $D(x,\theta_D)$, where x is the data sample and θ_D are the parameters of the D. The training process of a GAN is based on a minimax game between G and D networks and it is schematized in Figure 5.1. The minimax game is a concept derived from game theory where in a game between two players, one player tries to minimize the value (the loss in our case) that the other player can

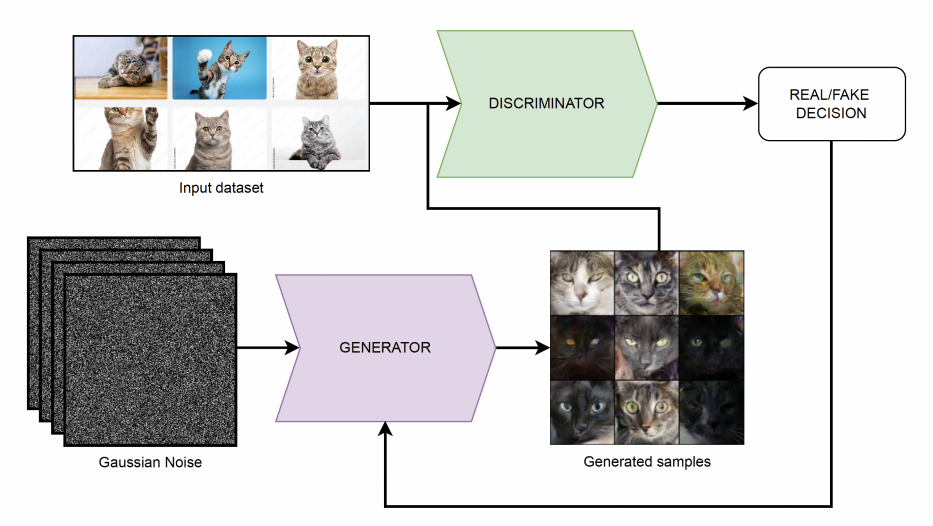


Figure 5.1: Generative Adversarial Network schematization. The Generator is trained to produce samples that mimic the data starting from noise. Then its output is fed alongside the real dataset to the discriminator which is trained in a supervised fashion to distinguish between the two. The decision made by the discriminator is then used as feedback to improve the Generator's capabilities.

force them to receive. In the case of the loss of the GAN is then formalized as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x,\theta_D)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z,\theta_G),\theta_D))] \quad (5.1)$$

where $V(D,G)$ is the value function of the GAN, $p_{\text{data}}(x)$ is the distribution of the training data, $p_z(z)$ is the distribution of the random noise vector, $D(x)$ is the output of the D, $G(z)$ is the output of the G network. We can think of this as an iterative process where the D discovers imperfections in the G and the Generator uses the feedback from the D to improve its performance. The training process is based on the idea of ideally finding a Nash equilibrium between G and Ds, another concept from game theory where each of the players cannot do better by unilaterally changing strategy. The end result is a situation where G generates data samples that are indistinguishable from the training data for D.

Mode Collapse

One of the main challenges of training GANs is mode collapse, where the generator learns to generate only a few modes of data distribution, instead of capturing the entire distribution. This can happen if the discriminator is too weak or has hit a local minimum and the generator finds a subset of the distribution that constantly fools D, leading G to generate the same samples. There are several techniques to prevent mode collapse, including minibatch discrimination, feature matching, and spectral normalization.

5.2 GAN variants

Since the introduction of GANs, there have been many variants and extensions proposed to improve the training stability and performance of GANs. Some of the most popular GAN variants include conditional GANs, Wasserstein GANs, and Least Squares GANs.

Conditional GANs

Conditional GANs [70] are a variant of GANs where both G and D are conditioned on some extra information y. That could be any kind of complementary information, from a label to a convoluted distribution. The objective function of a Conditional GAN is defined as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{(x,y)\sim p_{\text{data}}}[\log D(x,y,\theta_D)] + \\ + \mathbb{E}_{z\sim p_z(z),y\sim p_{\text{data}}(y)}[\log(1-D(G(z,y,\theta_G),y,\theta_D))] \quad (5.2)$$

This allows the generator to produce samples based on some condition, such as a label, a class, or some physical property.

Wasserstein GANs

Wasserstein GANs [71, 72] (WGAN) are a variant of GANs that use the Wasserstein distance described in Section 4.4 as the loss function. The WGAN value function is defined using the Kantorovich-Rubinstein duality:

$$V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] \quad (5.3)$$

where $D(x)$ is the output of the discriminator for real data, and $D(G(z))$ is the output of the discriminator (or even better the *critic* as it's not trained to classify) for generated data. The WGAN results in a critic whose gradient is better behaved than the discriminator in the original GAN, leading to more stable training and better performance and making the generator optimization an easier task.

Least Squares GANs

Least Squares GANs [73] (LSGAN) have been designed to tackle the problem of vanishing gradient in the "vanilla" GAN caused by the sigmoid cross entropy loss function to make the gradient vanish. The objective of the LSGAN is defined as:

$$\min_G \max_D V(D,G) = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[(D(x)-1)^2] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z)))^2] \quad (5.4)$$

where $D(x)$ is the output of the discriminator for real data, and $D(G(z))$ is the output of the discriminator for generated data. The benefit of this loss can be seen in two aspects. Unlike vanilla GAN, which causes almost no loss for samples that lie on the correct side of the decision boundary, but are still far from the original data,

LSGAN still punishes such samples even though they are correctly classified. As a result, the generator will produce samples that come closer to the decision boundary. Secondly, penalizing these samples relieves the problem of vanishing gradients as the loss function is only flat at one point, unlike the sigmoid function, thus reducing the vanishing gradients' criticality and providing a more stable training performance.

5.3 Flow Models

Flow models are a class of generative models that are based on the idea of transforming a simple distribution into a complex one with a series of invertible transformations that change the input data space into a latent space during training, and then transform them back into the input space during sampling. By applying repeatedly the change of variables rule, the data space distribution "flows" through the series of operations to the latent space distribution. Finally, one can obtain the prior distribution and for this reason, this type of algorithm is called "normalizing flow". Flow models are trained by maximizing the log-likelihood of the data under the flow process. In this section, the basic concepts of flow models will be introduced along with some of the most popular flow models, including normalizing flows and continuous normalizing flows.

Normalizing Flows

The most basic flow model is the normalizing flow schematized in Figure 5.2, which is a sequence of invertible transformations that map a simple distribution to a complex distribution. One can describe this process with the following equation:

$$z = \{f_K \circ f_{K-1} \circ ... \circ f_1\}(x) \tag{5.5}$$

where x is the input data, z is the latent representation, and f_i are the invertible transformations. The log-likelihood of the data under the flow process can be computed using the change of variables formula:

$$\log p(x) = \log p(z) + \log \left| \det \frac{\partial f}{\partial x} \right| \tag{5.6}$$

where $p(z)$ is the prior distribution in the latent space, and $\det \frac{\partial f}{\partial x}$ is the determinant of the Jacobian matrix of the transformation f. The log-likelihood can be maximized using gradient-based optimization methods, such as stochastic gradient descent.

Affine Coupling Layers

One of the most popular types of invertible transformations used in flow models is the affine coupling layer. This layer is a bijective transformation that splits the input data into two parts and applies an affine transformation to one part while leaving the other part unchanged. Let $x \in \mathbb{R}^d$ be the input data, and $x_1 \in \mathbb{R}^{d_2}$ and $x_2 \in \mathbb{R}^{d_2}$ be the split components of the input data. The affine coupling layer transforms x_2 using x_1 as follows:

$$y_1 = x_1, \quad y_2 = x_2 \odot \exp(s(x_1)) + t(x_1) \tag{5.7}$$

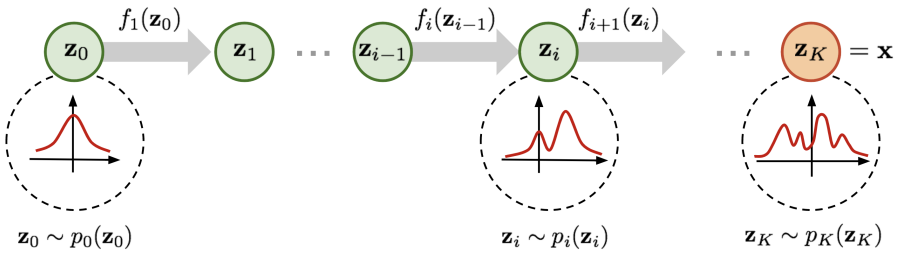


Figure 5.2: Normalizing Flow schematization. The input data is transformed into a latent space using a series of invertible transformations. During sampling, the latent space distribution is then transformed back into the input space. Image from [74]

where $s(x_1) \in \mathbb{R}^{d_2}$ and $t(x_1) \in \mathbb{R}^{d_2}$ are the scaling and translation functions of x_1 parametrized by neural networks, and $\odot$ is the element-wise multiplication. The operation is easily invertible:

$$x_1 = y_1, \quad x_2 = (y_2 - t(y_1)) \odot \exp(-s(y_1)) \tag{5.8}$$

and the Jacobian matrix is not hard to get and is given by the lower triangular matrix:

$$J = \begin{bmatrix} I & 0 \\ \frac{\partial y_2}{\partial x_1} & \operatorname{diag}(\exp(s(x_1))) \end{bmatrix} \tag{5.9}$$

for which the determinant is computed as the product of its diagonal elements:

$$\det \frac{\partial f}{\partial x} = \exp(\sum_i s_i(x_1)) \tag{5.10}$$

To ensure that all inputs undergo some alteration, the ordering of the dimensions is shuffled at each layer. This is done to avoid the problem of the model ignoring the dimensions of the input data.

Continuous Normalizing Flows

Instead of a discrete sequence of transformations, it is possible to expand the idea of normalizing flows by considering a continuous sequence of transformations where the latent space can be defined as a time-dependent function. Its evolution can then be described by an ordinary differential equation:

$$\frac{dz(t)}{dt} = v_t(z(t), t; \theta) \tag{5.11}$$

where v_t is a time-dependent vector field that defines the flow of the data through the latent space. This is parametrized by theta and modeled with a neural network. Equation (5.11) is known as the *neural ordinary differential equation* [75]. The vector field defines a probability density path p_t of the latent space, describing how $z(t)$ evolves over time. At time $t = 0$, $p_0(z_0)$ is the prior distribution, and at time $t = 1$,

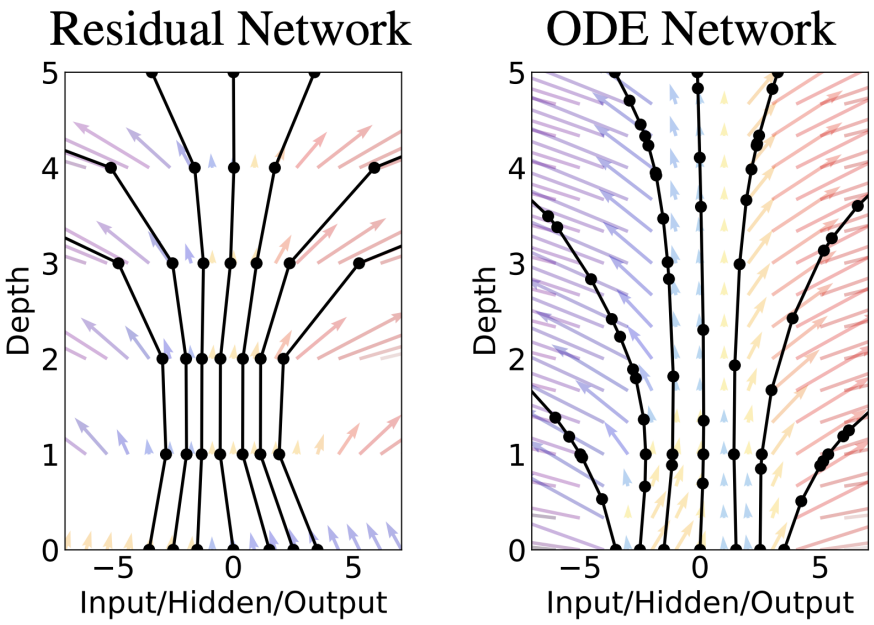


Figure 5.3: Comparison of normalizing flow (left) to continuous normalizing flow (right). Normalizing flow defines a discrete set of transformations, while the ODE defines a vector field that continuously changes the state. Taken from [75]

$p_1(z_1)$ is the distribution of the data. The log-likelihood of the data under the flow process can be computed using the change of variables formula:

$$\log p(x) = \log p(z) + \int_0^1 \nabla \cdot v_t(z(t))dt. \tag{5.12}$$

Once the model is trained, it is possible to sample from the base distribution $p_0(z_0)$, typically a standard normal distribution, and solve the ODE to obtain the data distribution. Figure 5.3 shows the schematization of a continuous normalizing flow vector field compared to a discrete normalizing flow.

Flow Matching

Flow matching [76] introduces a shift in the training paradigm of continuous normalizing flows.
Let z be a random variable with an unknown distribution $q_{\text{data}}(z)$. It is possible to define a time-dependent distribution $p_t(z)$ such that at time $t = 0$ p_0 matches a simple distribution, e.g. a standard normal distribution, and at time $t = 1$, p_1 is a distribution that closely approximates $q_{\text{data}}(z)$. We can then deterministically obtain the vector field u_t that generates p_t. Given now this vector field u_t, we can define the Flow Matching loss function as:

$$L(\theta) = \mathbb{E}_{t\sim\mathbb{U},p_{t(z)}}[\|v_t(z) - u_t(z)\|^2] \tag{5.13}$$

where θ are the parameters of the neural network that defines the vector field v_t, and $z \sim p_{t(z)}$. The meaning of the loss is that we regress the vector field u_t with the

neural network v_t and once the loss is minimized, the vector field v_t will generate the distribution p_t that approximates the data distribution $q_{\text{data}}(z)$. Ref. [76] shows that it is possible to construct p_t and u_t from *conditional probability paths and vector fields.*

5.4 Diffusion Models

Diffusion models have grown popular in recent years in the field of generative machine learning for tasks such as image generation and data augmentation. The core idea is to train a network by gradually perturbing the input data until they follow a normal distribution and then learn the transformation to reverse the perturbation. In this section, we will introduce the basic concepts of diffusion models and some of the most popular diffusion models, including denoising diffusion probabilistic models, and k-diffusion.

Denoising Diffusion Probabilistic Models

The first concept of a diffusion model was proposed by Sohl-Dickstein et al. [77] in 2015 and is based on the idea of building a Markov chain that gradually converts one distribution to another. In more technical terms, we can consider a random variable $z_0 \sim q(z_0)$ and latent variables $z_1, \ldots, z_T$, where the index t represents the time step and T, is the total number of steps.

The joint distribution q_θ is refered to as *reverse process* or, in other words, the process of removing the noise from the input data. This is defined as a Markov chain of learned Gaussian transition and is described as:

$$q(z_{0:T}) = q(z_T) \prod_{t=1}^{T} q_\theta(z_{t-1}|z_t) \tag{5.14}$$

where:

$$q_\theta(z_{t-1}|z_t) = \mathcal{N}(z_{t-1}; \mu_\theta(z_t, t), \Sigma_\theta(x_t, t)) \tag{5.15}$$

describes every link in the chain that starts at $q(z_t) = \mathcal{N}(z_t; 0, \mathcal{J})$. μ_θ and Σ_θ are the mean and covariance of the Gaussian transition that is parametrized by a neural network with parameters θ that take as input the data z_t and the time step t with $1 \leq t \leq T$. On the other hand, in the diffusion models *forward process* the posterior $q(X_{1:T|x_0})$ is bounded to a Markov chain that adds Gaussian noise at each step with a well-defined variance schedule $\beta_1, \ldots, \beta_T$:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{5.16}$$

where:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbb{I}) \tag{5.17}$$

The training of the model is done by optimizing the variational bound on negative log-likelihood, also known as ELBO:

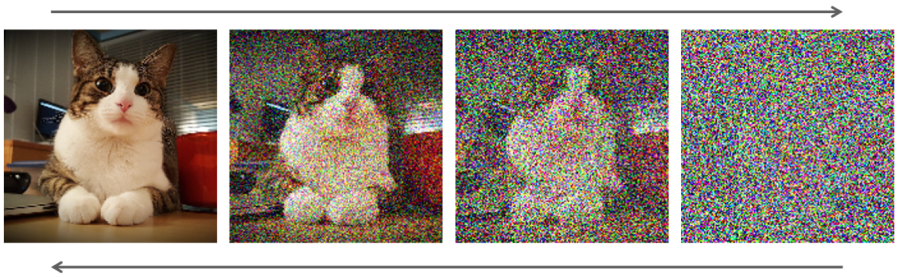


Figure 5.4: Shown is a schematic view of the diffusion process during training (arrow pointing to the right) and sampling (arrow to the left). During training the picture is gradually perturbed to obtained a progressively more noisy image and at eache step the model is trained to learn how to recognise the added noise and reverse the process. During sampling, the trained models starts with noise and step by step removes the noise to produce a sample from the target distribution. Taken from [78]

.

$$\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right] = \\ = \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right] =: L \quad (5.18)$$

Sampling at any time step t is allowed in closed form in the forward process. Using the notation $\alpha_t := \sqrt{1-\beta_t}$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$ we obtain:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_0, (1-\bar{\alpha}_t)\mathbb{I}) \quad (5.19)$$

A schematic view of the training and sampling processes is shown in Figure 5.4

Score Matching

The DDPM model just introduced is based on the idea of denoising the input with a fixed number of steps. One can also consider the idea of denoising the input with an infinite number of steps, which leads us to the idea behind the score-matching method also known as *k-diffusion*. Score matching [79] is a method for estimating the score function of a probability distribution. The denoising process can be described by stochastic differential equations (SDE). The score function is the gradient of the log-likelihood of a probability distribution for the data $\mathbf{x}$:

$$s(x) = \nabla_x \log p(x). \quad (5.20)$$

With this knowledge, it is possible to model the original data density and hence produce new samples by approximating s with a neural network. This has the quality

to change the goal of the training to a simpler task in which we can regress the score function of the data distribution to the score function of the model distribution using the standard MSE loss:

$$L_{\mathrm{SM}}(\theta) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}\,(\mathbf{x})}\left[\left\|\mathbf{s}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\mathrm{data}}\,(\mathbf{x})\right\|^2\right]. \tag{5.21}$$

Unfortunately, the score for the data is not known a priori, but one can obviate this by using a predetermined noise distribution $q_\sigma(\hat{\mathbf{x}}|\mathbf{x})$ to perturb the data so that score matching can be used to estimate the score of the polluted data distribution $q_\sigma(\hat{\mathbf{x}}) = \int q_\sigma(\hat{\mathbf{x}} \mid \mathbf{x}) p_{\mathrm{data}}\,(\mathbf{x}) d\mathbf{x}$. The denoising score matching function can then be written as:

$$L_{\mathrm{SM}}(\theta) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}\,(\mathbf{x}),\hat{\mathbf{x}}\sim q_\sigma(\hat{\mathbf{x}}|\mathbf{x})}\left[\left\|\mathbf{s}(\hat{\mathbf{x}}) - \nabla_{\hat{\mathbf{x}}} \log q_\sigma(\hat{\mathbf{x}} \mid \mathbf{x})\right\|^2\right]. \tag{5.22}$$

The most common choice for the noise distribution is the Gaussian distribution $q_\sigma(\hat{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{x}}; \mathbf{x}, \sigma^2 \mathbb{I})$ that allows for the score function to be computed in closed form assuming $\epsilon = \hat{\mathbf{x}} - \mathbf{x}$ can be sampled from the standard normal distribution:

$$\nabla_{\hat{\mathbf{x}}} \log q_\sigma(\hat{\mathbf{x}} \mid \mathbf{x}) = -\frac{\epsilon}{\sigma^2}. \tag{5.23}$$

The score matching loss can then be computed as:

$$L_{\mathrm{SM}}(\theta) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}\,(\mathbf{x}),\epsilon\sim\mathcal{N}(0,\mathbb{I})}\left[\left\|\mathbf{s}(\mathbf{x} + \sigma\epsilon) + \frac{\epsilon}{\sigma^2}\right\|^2\right], \tag{5.24}$$

which means that we are computing the MSE between the output score of the model and the noise vector ϵ.
Training has proved to be very stable and has shown to be able to generate high-quality samples as we will see in Section 9 where we will introduce the application of such model to the very challenging setup provided by the CMS HGCal.

6 Shared Data and Algorithms for Physics Data

High-quality training data are a critical resource for progress in machine learning, with curated benchmark datasets facilitating reliable comparisons and sustained advancements. Renowned benchmarks like MNIST [80], CIFAR [81], and ImageNet [82] have significantly contributed to the success of deep learning in image processing. Similarly, fundamental physics relies on datasets and challenges [83–86], which drive continuous progress.

Physics data poses unique challenges due to diverse representations stemming from varying experimental designs and theoretical frameworks. Representations include high-level observables [83], images [87], sequences [88], point clouds [89], graphs [90], and hybrid formats [91]. While domain-specific approaches are feasible, they lead to inefficiencies and redundant efforts. Instead, flexible architectures compatible with multiple data sources can enhance the accessibility and utility of deep learning in scientific applications.

This chapter will describe the PD4ML PYTHON package introduced in 2021, which unifies datasets across experimental and theoretical domains, to which the writer of this thesis was a main contributor by both setting up the provided datasets, writing a big part of the codebase, and performing training and evaluations to verify the claimed gain of the work. This work lead to a publication [92] upon which this chapter is based.

Unlike existing domain-specific collections, PD4ML simplifies access through a consistent, ready-to-use interface, minimizing preprocessing requirements. It focuses on supervised learning tasks, providing clear performance metrics such as accuracy and AUC for classification and resolution for regression tasks. All datasets are accompanied by reference implementations of state-of-the-art algorithms.

Included datasets span diverse physics domains:

- Top tagging in LHC simulations [85];
- Event selection in Belle II simulations [93];
- Phase transition detection in nuclear collisions [94];
- Phase transition classification with domain adaptation [95];
- Shower maximum reconstruction in cosmic-ray observatories.

These datasets cover particle physics, flavor physics, hadronic and nuclear physics, and astroparticle physics. Contributions of new supervised learning tasks are encouraged and supported via a streamlined submission protocol [1].

Given the diversity of physics data, the choice of representation and network ar-

[1] Details available at `https://github.com/erum-data-idt/pd4ml`

Table 6.1: Overview of the provided datasets.

	Task	Examples (train/test/validation)	Structure	Dimension
Top Tagging Landscape	Class.	1.2M/400k/400k	Four vectors	200 particles, 4 features/particle
Smart Backgrounds	Class.	157k/39k/84k	Decay Graph	100 particles, 9 features/particle
Spinodal or Not	Class.	16.3k/4k/8.7k	2D Histogram	20x20 histogram of pion spectra
EoS	Class.	121k/25k/54k	2D Histogram	24x24 histogram of pion spectra
Air Showers	Regr.	56k/30k/14k	81 1D Traces	81 stations, 80 signal bins + timing

chitecture is pivotal. Fully-connected networks (FCNs) are broadly applicable but lack efficiency and risk overfitting, while problem-specific architectures require fine-tuning. A middle ground, such as graph-based representation, was chosen as it was an emerging technology that could balance generalizability and performance. Graphs, composed of nodes and edges, enable message-passing mechanisms for processing data, as elaborated in Section 6.3. Each dataset includes a strategy for graph representation.

The chapter is structured as follows: datasets and tasks are briefly introduced in Sec. 6.1, the Python interface is detailed in Sec. 6.2, and model comparisons between fully-connected and graph-based architectures are discussed in Sec. 6.3. A summary and outlook are presented in Sec. 6.4, with reference architectures provided for all tasks. As this work has already been published, the following sections are based on the original publication [92], and some of the images can be the same as in the original publication.

6.1 Datasets

An overview of the included datasets is given in Table 6.1 and in the following, additional information on the physics challenge, data generation process, and other details are given for all datasets.

Top Tagging Landscape

Tagging particles based on their decay products is a key task at the LHC, especially for identifying hadronically decaying top quarks with high Lorentz boosts, which are crucial in exploring physics beyond the Standard Model.

The top tagging dataset [96], designed for benchmarking classification algorithms [85], allows direct comparisons between general and state-of-the-art methods. It includes jets generated with Pythia [97], simulated using Delphes [56], and reconstructed with the Anti-k_T algorithm [98] in FastJet [99]. The dataset provides the 200 constituent four-vectors of the highest transverse momentum jet per event, with zero-padding for smaller jets, comprising 1.2M training and 400k testing and validation examples. Further details are in Ref. [85].

Smart Backgrounds

Simulation poses significant computational challenges for experiments like Belle II, which require large-scale simulated datasets alongside measured data. The simulation process includes fast event generation, simulating decay chains, and resource-intensive detector simulation and reconstruction. Background generation must account for all possible decay chains, but stringent selection criteria typically retain only a tiny fraction of simulated background events—often as low as 10^{-7}. This results in considerable wasted computing resources.

Efficiency can be improved by filtering events after generation, a step that uses only about 0.1% of the total simulation time. Predicting which events will pass final selection without full simulation is a classification problem initially tackled using CNNs [100] and later enhanced with graph neural networks [93].

The *Smart Backgrounds* dataset includes simulated $e^+e^- \to \Upsilon(4S) \to B^0\bar{B}^0$ decays generated with EvtGen [101]. Features include particle four-momentum, production vertex positions and time, particle types (mapped via PDG identifiers [102]), and mother-particle indices, enabling graph representation of decay trees. Events contain up to 100 particles, with zero padding and default values for shorter decay chains.

Labels indicate whether an event passes filtering criteria based on the Full Event Interpretation (FEI) algorithm [103], which reconstructs hadronic B decays post-detector simulation. The FEI's broad applicability results in a relatively high retention rate of 5%, meaning one in 20 simulated background events is labeled as passed.

Spinodal or Not

The *spinodal* dataset [104] is a simulated dataset designed to study the effects of non-equilibrium deconfinement phase transitions in relativistic nuclear collisions, providing insight into the strong interaction in high baryon density regimes. These investigations are crucial for understanding QCD phenomena like deconfinement and mass generation. The Compressed Baryonic Matter (CBM) experiment at FAIR (GSI Darmstadt) aims to explore these properties by colliding heavy nuclei (e.g., lead or uranium) at several GeV per nucleon, creating short-lived equilibrated systems with high-density and temperature.

The dataset is based on fluid dynamical simulations of heavy-ion collisions under two scenarios: one with spinodal decomposition (characterized by phase separation and exponential density fluctuation growth) and one without. The aim is to identify events with spinodal decomposition and understand their distinguishing features compared to non-spinodal events. High classification accuracy is critical for determining whether all simulated spinodal events exhibit the expected characteristics.

The dataset contains 27,000 central lead-on-lead collision events at a beam energy of $E_{\text{lab}} = 3.5\,A$ GeV for each scenario. Each event is represented as a 20×20 pixel histogram of the net baryon density distribution in the transverse X-Y plane, normalized per event to mitigate artifacts. These histograms are flattened into 400-column arrays for classification tasks. Further details on the physics motivation and methodology are available in Ref. [94, 105].

EoS

The EoS dataset is based on relativistic hydrodynamic simulations, including hadronic cascade afterburners, to study high-energy heavy-ion collisions and investigate the QCD phase structure. Specifically, these collisions aim to locate the critical endpoint (CEP) in the QCD phase diagram, which separates crossover transitions from first-order phase transitions between hadronic matter and quark-gluon matter. Experiments at RHIC, LHC, FAIR, and NICA provide abundant data, but traditional methods of identifying critical fluctuations face challenges due to weak signals and complex physical factors.

This dataset reframes the problem as a deep learning classification task to explore the nature of the QCD transition encoded in the equation of state (EoS) used in the hydrodynamic simulations. Simulations were performed with the iEBE-VISHNU hybrid model using event-by-event hydrodynamics combined with MC-Glauber and MC-KLN initial condition generators. Two EoS types are considered: a crossover EoS from lattice QCD and a first-order EoS with Maxwell construction. The simulations include stochastic particleization, resonance decays, and hydronic cascades via UrQMD, with varying physical parameters to ensure robustness and reduce biases.

Each event in the dataset is labeled as either EOSL (crossover EoS) or EOSQ (first-order EoS) and represented as a 2D histogram of pion spectra with 24 transverse momentum bins and 24 azimuthal angle bins. This format serves as the input for deep learning models. Previous studies demonstrated the ability of CNNs to classify the EoS with high accuracy and robustness against other physical variations [106–108]. This dataset facilitates further exploration of QCD phase transitions through machine learning.

Cosmic-ray induced Air Showers

When ultra-high-energy cosmic rays (UHECRs) enter Earth's atmosphere, they produce extensive air showers, which can be detected by ground-based observatories using water-Cherenkov detectors or scintillators. Each detector records time-dependent particle densities as signal traces, which encode information about the shower's development. Of particular interest is X_{max}, the depth of the shower maximum, which provides insights into the cosmic-ray mass. While X_{max} is typically reconstructed using fluorescence telescopes, these instruments have limited duty cycles, making it challenging to gather sufficient data. Ground-based particle detectors, with their higher uptime, could greatly improve the statistics for X_{max} measurements.

The air-shower dataset [109] enables X_{max} reconstruction using simulations of a ground-based observatory. The simulated setup is inspired by the Pierre Auger Observatory and Telescope Array Project, featuring a Cartesian array of detectors spaced 1500 m apart at an altitude of 1400 m. Events are modeled using a fast-simulation approach [110], with parameterized and simplified shower developments.

To optimize memory usage, the air-shower footprint is reduced to a 9×9 detector array centered on the station with the largest signal. Each station's data includes time-dependent signal traces over 80 time steps (25 ns per step) starting from the first detected signal. Additionally, the arrival time of the first particles is recorded for

each station. This data structure forms a 2D spatial arrangement with a temporal dimension, resembling a 3D input for machine learning.

The task is to perform regression to predict $X_{\max}$ from the detector readouts with high resolution, defined as the standard deviation of the differences between predicted and true $X_{\max}$ values.

6.2 Python Interface

As it is one of the most commonly used code languages in the machine learning community, we chose PYTHON as the primary language for the interface of the PD4ML package.

With minimal coding effort, users can load any of the five datasets, apply dataset-specific preprocessing, and generate adjacency matrices tailored for graph neural network algorithms. The package is designed to be highly extensible, allowing additional datasets to be integrated seamlessly through custom wrappers.

The core function, `load`, facilitates the loading of training and testing datasets. The dataset features ($\mathbf{X}$) and labels ($\mathbf{y}$) are returned as NUMPY arrays [111], ensuring compatibility with most Python-based machine learning frameworks. Below is an example demonstrating how to load the training and testing sets for the Spinodal dataset:

```
from pd4ml import Spinodal
X_train, y_train = Spinodal.load('train')
X_test, y_test = Spinodal.load('test')
```

If the specified dataset path is not found, the data is automatically downloaded. A forced download option is also available, and an MD5 checksum is implemented to ensure data integrity.

Additionally, the package provides a `load_data` method, which applies preprocessing routines specified by the dataset providers. Users can load the data in graph form by setting the `graph` argument to `True`. The resulting adjacency matrices are constructed as detailed in Sec. 6.3. An example is shown below:

```
X, y = Spinodal.load_data('train',
                          path='.',
                          graph=True)
```

The package also includes functionality to display a dataset's description by executing the following command:

```
Spinodal.print_description()
```

The output contains both technical details about the dataset (e.g., the number of events) and references to the relevant scientific literature, providing context on the underlying physics and reference models.

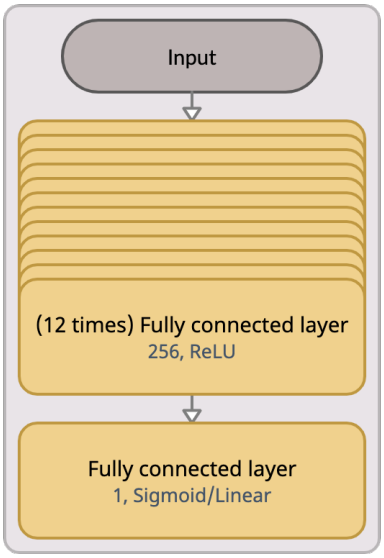


Figure 6.1: Schematic representation of the FCN. Additional details are provided in the text. For classification tasks Sigmoid is used as loss in the final layer, for regression a linear activation is used instead.

Furthermore, the repository [2] includes a TensorFlow 2.3 [112] implementation of the reference models described in Sec. 6.3 and the standard models introduced in Sec. 6.3.

6.3 Example Application

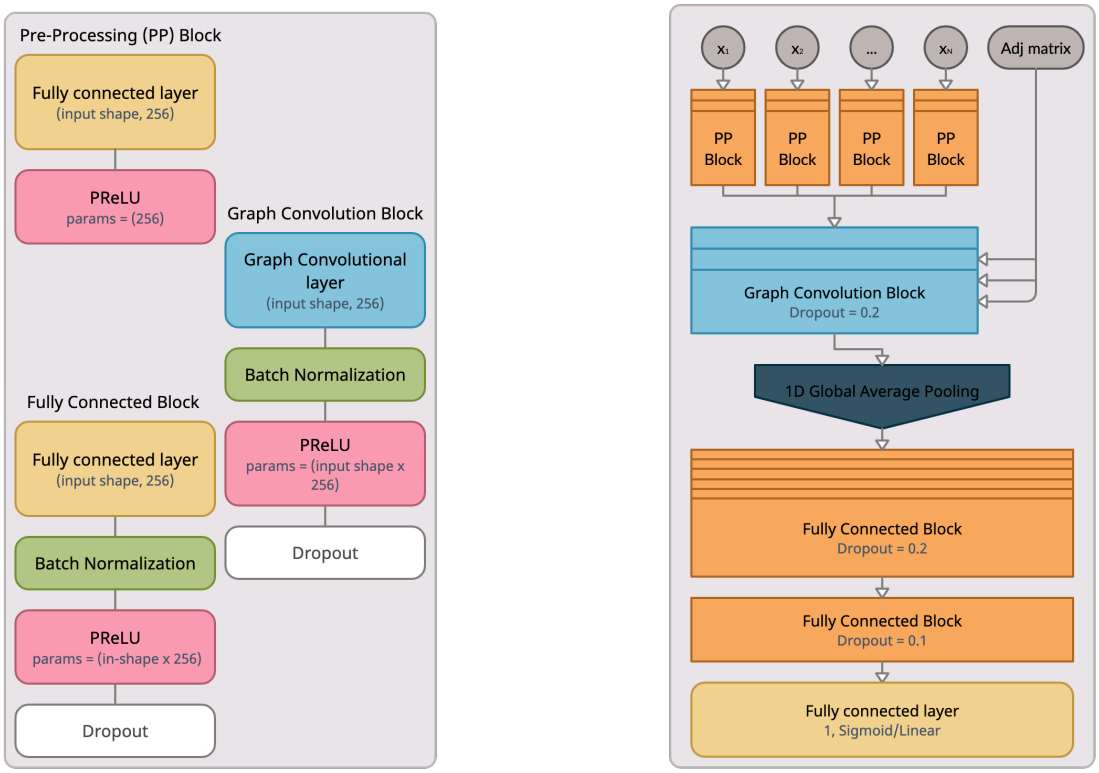
To showcase the utility of a standardized interface for tasks such as transfer learning, two widely used architectures are considered: Fully Connected Networks (FCNs) (Sec. 6.3) and Graph Neural Networks (GraphNets) (Sec. 6.3). Both architectures are trained on all datasets, with the primary difference being adjustments to the number of input layers specific to each dataset.

Each dataset is also accompanied by a reference model, designed to provide a baseline for performance comparisons. These models are outlined in Sec. 6.3. For established datasets, like the Top Tagging Landscape task, state-of-the-art models are utilized as benchmarks. For newer datasets, reference implementations are developed and provided by their creators.

Fully-Connected Network

We employ a foundational Fully Convolutional Network (FCN) architecture for our experiments across all datasets (Sec. 6.1). This choice eliminates the need for prior assumptions about data structure, enhancing model versatility.
Our FCN comprises 12 hidden layers, each with 256 nodes and ReLU activation. While an exhaustive hyperparameter search was not conducted, variations in layer

[2] `https://github.com/erum-data-idt/pd4ml`



(a) Constituent blocks of the GraphNet

(b) GraphNet

Figure 6.2: Schematic representation of the GraphNet model. Additional details are provided in the text.

count, batch size, and learning rate demonstrated minimal impact on results. The network's output layer and loss function are task-specific:

- **Classification:** Sigmoid output and Binary Cross Entropy loss.
- **Regression:** Linear output and Mean Squared Error loss.

Training involves 256 example batches over 300 epochs. To mitigate overfitting, early stopping halts training after 15 consecutive epochs without validation loss improvement.

The Adam optimizer [113] with an initial learning rate of 0.001 drives the training process. The learning rate is dynamically adjusted, decreasing by a factor of 10 after 8 epochs without validation loss improvement.

Graph-based Network

Data can often be naturally represented as graphs or transformed into graph structures without information loss. This approach is particularly applicable in fundamental physics, where measurement signals can be represented as graph nodes and their interrelations or the relations between measurement devices as edges. Unlike fully connected networks (FCNs), this representation leverages additional information about distances and connectivity, which FCNs cannot inherently process.

For instance, particle jet events involve multiple particles whose order is permutation invariant but are associated with spatial locations and additional features like energy, time, and particle type. While jet data is often pre-processed into a 2D image representation, directly using a graph-based neural network can improve classification performance [85].

In this work, we consider undirected graphs with N nodes, where an edge between nodes i and j is equivalent to an edge between j and i. Each node is assigned a feature vector $n_i \in \mathbb{R}^M$, where M is the dimension of the feature space. For simplicity, we focus on unweighted graphs, although edge features could also be incorporated. Each graph is described by two matrices: the feature matrix $X \in \mathbb{R}^{M \times N}$ and the adjacency matrix $A \in \{0,1\}^{N \times N}$. An entry of 1 in A indicates a connection between nodes i and j, while 0 indicates no connection. A dataset is composed of multiple such graphs, with each data point represented by a feature matrix and an adjacency matrix. Nodes and edges are stored separately, with node features corresponding to standard data features and edge connections encoded in the adjacency matrix.

Many data types can be transformed into graphs without losing information. For instance, image data can be represented as a graph by constructing an adjacency matrix based on pixel neighbors as shown later in this Section. If absolute pixel positions are relevant, they can be included as per-node features.

Our GraphNet implementation follows a structure similar to the Smart Backgrounds reference model described later in thisSsection. Input data are processed in batches of 32, with training conducted over 400 epochs and regulated by early stopping with patience of 50 epochs. The Adam optimizer is used, starting with a learning rate of 0.001, which decreases by a factor of 10 if the validation loss does not improve within 8 epochs.

The key distinction between our GraphNet and an FCN is the inclusion of an adjacency matrix, which serves as input to graph convolutional layers [114]. The architecture includes three fully connected layers per node with shared weights across nodes, followed by three graph convolutional layers, a 1D global average pooling operation, three additional fully connected layers, and an output layer. Batch normalization and dropout [115] (0.2 for most layers and 0.1 for the last layer) are applied after each graph convolutional layer. All layers use 256 trainable nodes and PReLU activations [116], except the output layer, which employs a sigmoid or linear activation depending on the task (classification or regression). A schematic of the architecture is shown in Fig. 6.2. Preprocessing follows the same steps as for the FCN, with an additional dataset-specific step to construct adjacency matrices, as detailed in the next section.

Pre-Processing and Adjacency Matrix

For each dataset, basic preprocessing is applied, closely following the corresponding reference model methods.

- **Top Tagging Landscape Dataset**: Data is transformed from four vectors to four hadronic coordinates: the logarithm of the transverse energy ($\log(\mathrm{p})_\mathrm{T}$), the logarithm of the energy ($\log(\mathrm{E})$), the relative pseudorapidity ($\Delta\eta$), and the relative azimuthal angle ($\Delta\phi$).

Table 6.2: Summary how different structures can be represented as graphs

Dataset	Structure	Graph building
Top Tagging Landscape	Four vectors	k-nearest neighbor clustering
Smart Backgrounds	Decay Graph	(not needed)
Spinodal or Not	2D Histogram	adjacent pixels
EoS	2D Histogram	adjacent pixels
Air Showers	81 1D Traces	geometric relation of detector stations

- **SmartBKG Dataset**: Particle ID information is one-hot encoded, while other features remain unchanged.
- **EoS Dataset**: Data is standardized.
- **Air-Shower Dataset**: Preprocessing involves taking the logarithm of the filled signal bins and normalizing timing values.
- **Spinodal or Not Dataset**: No preprocessing is performed.

The construction of adjacency matrices is tailored to the characteristics of each dataset:

- Top Tagging Landscape Dataset (Sec. 6.1): A k-nearest neighbor clustering ($k = 7$) is performed using information from the jet constituents.
- Smart Background (BKG) Dataset: Generator-level particles and their mother-daughter relationships are used to construct the adjacency matrix.
- EoS and Spinodal Datasets: Both datasets are represented as 2D histograms. The adjacency matrix is built by counting the eight neighboring bins for each "pixel" (three at corners and five at edges).
- Air-Shower Dataset: Detectors are arranged in a 9×9 rectangular grid, allowing adjacency matrix construction based on the same eight-adjacent-bins technique.

A summary of these graph-building methods is presented in Table 6.2.

Reference Models

To assess the performance of our dataset-independent algorithms, we compare them against reference models. These models, tailored to each dataset, represent the best achievable performance for the specific problem.

Top Tagging Landscape

The ParticleNet algorithm [117] serves as the reference model for the Top Tagging Landscape dataset. This architecture was among the top-performing models in a prior comparison study on this dataset [85].

ParticleNet employs a graph convolutional network (GCN) by treating the input data as point clouds, where each jet is represented as an unordered set of particles. To construct these point clouds, particles within each jet are sorted by transverse momentum and zero-padded to a maximum of 100 particles per jet. Seven input

features are derived from the particles' four-momenta:

1. The logarithm of the transverse energy ($\log(p_{\mathrm{T}})$),
2. The logarithm of the energy ($\log(E)$),
3. The relative pseudorapidity ($\Delta\eta$),
4. The relative azimuthal angle ($\Delta\phi$),
5. The logarithm of the particle's transverse momentum relative to the jet's transverse momentum ($\log(p_{\mathrm{T}}/p_{\mathrm{T}}^{\mathrm{jet}})$),
6. The logarithm of the particle's energy relative to the jet's energy ($\log(E/E^{\mathrm{jet}})$), and
7. The angular separation between the particle and the jet axis defined as $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$.

Relative angles are calculated with respect to the jet axis. Using these inputs, a graph is constructed for each jet by performing a standard k-nearest neighbor search ($k = 7$) to determine the closest particles.

The ParticleNet architecture consists of three EdgeConv layers [118], followed by a global pooling operation across all particles to ensure permutation invariance, and concludes with two fully connected layers.

Smart Backgrounds

The event decay tree's graph structure is particularly well-suited for graph neural networks (GNNs), with graph convolutional layers [114] enabling high classification performance.

In this approach, particle PDG identifiers are passed through an embedding layer to generate an 8-dimensional embedding, which is concatenated with an additional 8-dimensional feature vector. The resulting 16-dimensional particle features are input into three fully connected layers with shared weights across all particles, followed by ReLU activations.

Subsequently, three graph convolutional layers process the data, utilizing an adjacency matrix constructed from the indices of mother particles. This matrix is symmetrized to account for both mother-daughter and daughter-mother relationships. The output is reduced to event-level quantities by averaging the resulting vectors across the particle dimension.

These event-level features are further processed through three additional fully connected layers with ReLU activations and a final fully connected layer with a sigmoid activation function to produce the classification score. All hidden layers, including the graph convolutional layers, comprise 128 units.

Spinodal or Not

Given the 20x20 histogram format of the simulated output data, we utilize a convolutional neural network (CNN) architecture. This network comprises three convolutional layers interspersed with pooling layers, followed by a fully connected hidden layer and

the output layer. The complete structure of the network is shown in Ref.[94].

EoS

Motivated by the success of CNNs in image recognition, we designed a VGG-like CNN architecture [119] with three convolutional layers followed by fully connected layers for EoS binary classification. The input to the CNN is an "image" derived from the pion histogram at minimum rapidity. To mitigate overfitting, we incorporated batch normalization, dropout, and PReLU activation. For detailed network architecture and training procedures, please refer to Ref. [107]. Further technical explanations can be found in Ref. [106].

Cosmic-ray induced Air Showers

To process the time- and space-dependent air-shower footprints, the model employs a two-part architecture inspired by Refs. [110, 120].
The first component is a recurrent network designed to analyze the signal traces recorded at each detector station. This component consists of a two-layer subnetwork of Long Short-Term Memory (LSTM) networks [121], shared across all 9×9 detector stations. Each signal trace is processed to extract ten features per station.

The resulting output has an image-like structure ($9 \times 9 \times 10$) and is concatenated with a map of arrival times ($9 \times 9 \times 1$) to capture spatial correlations. These combined feature maps are analyzed using convolutional operations in two blocks, separated by a pooling operation. Each block contains five residual units [122] with convolutional layers and batch normalization [123]. Following the residual blocks, global max pooling and dropout are applied.

To accelerate model convergence, a re-normalization layer adjusts the outputs, initially scaled between 0 and 1, to match the scale of the X_{max} distribution.

For the reference architecture, preprocessing involved logarithmic rescaling of signal traces to address variations in signal size and normalizing arrival times relative to the time measured at the central station (the station with the largest signal). Further details can be found in Ref. [110].

Performance Evaluation

Model performance was assessed by training each model five times with identical data but different random initializations. The mean and standard deviation of the resulting metrics were then computed. For classification tasks, performance was evaluated using *accuracy* (the proportion of correctly classified samples) and the *area under the curve* (AUC). For regression tasks, the *resolution*, as described in Sec. 6.1, served as the primary metric.
Tables 6.3 and 6.4 provide an overview of the accuracy and AUC results, respectively, while Fig. 6.3 illustrates relative accuracy compared to the reference model.
The GraphNet delivered performance on par with or close to the reference model for classification tasks. The largest disparity occurred with the Spinodal dataset,

Table 6.3: Accuracy scores of the different models. Provided are the mean value and its standard deviation of five independent pieces of training on the same data.

	Reference	GraphNet	FCN
TopTag	0.940 ± 0.001	0.935 ± 0.001	0.908 ± 0.001
SmartBkg	0.823 ± 0.001	0.824 ± 0.001	0.737 ± 0.002
Spinodal	0.873 ± 0.004	0.854 ± 0.004	0.824 ± 0.001
EoS	0.691 ± 0.005	0.687 ± 0.005	0.605 ± 0.019

Table 6.4: AUC scores of the different models. Provided are the mean value and its standard deviation of five independent pieces of training on the same data.

	Reference	GraphNet	FCN
TopTag	0.986 ± 0.001	0.983 ± 0.001	0.966 ± 0.002
SmartBkg	0.906 ± 0.009	0.903 ± 0.001	0.811 ± 0.001
Spinodal	0.925 ± 0.005	0.916 ± 0.005	0.883 ± 0.001
EoS	0.788 ± 0.005	0.766 ± 0.005	0.739 ± 0.008

where the GraphNet trailed the reference model by 2% in AUC and 1% in accuracy. Compared to the fully connected network (FCN), however, GraphNet demonstrated considerably better results, with a notable 12% advantage on the EoS dataset.
Regression outcomes for the air-shower dataset (Tab. 6.5) revealed more significant gaps. The GraphNet showed a 10% decline in resolution compared to the reference model, while the FCN performed worse by 30%.

6.4 Outlook and Conclusions

The results show that a unified graph-based architecture could achieve near state-of-the-art performance across diverse tasks using shared hyperparameters. This suggested the proposed architecture[3] could serve as a reliable baseline — or even a default choice — for machine learning in fundamental physics. The outlined graph construction methods also provide a flexible framework for adapting various data structures.

[3]Implementation available at `https://github.com/erum-data-idt/pd4ml`

Table 6.5: MSE and resolution values measured on the air-shower dataset. Provided are the mean value and its standard deviation of five independent pieces of training on the same data.

Air Shower	MSE	Resolution
Ref. Model	1000 ± 52	31.32 ± 0.75
GraphNet	1185 ± 26	34.12 ± 0.47
FCN	1661 ± 19	40.63 ± 0.40

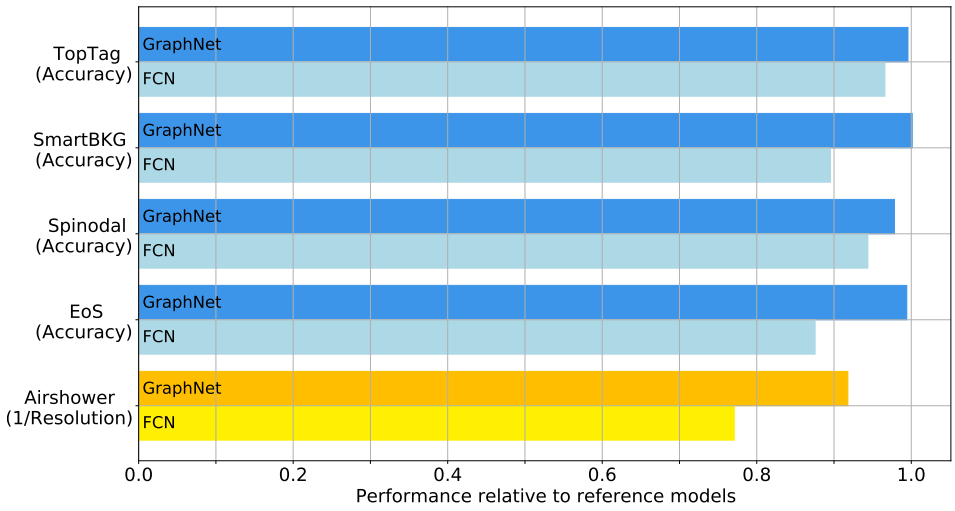


Figure 6.3: Performance of the GraphNets and the FCNs relative to the reference models for the different datasets.

However, the performance gap observed for cosmic-ray air-shower data underscores the need for further improvements, especially for complex datasets. At the time this thesis is being written, there are probably more advanced architectures available that tackle the question of transfer learning. However, this work was one of the earliest to address the question of transfer learning in the context of fundamental physics to the knowledge of the writer.

7 Dynamic Graph Neural Networks for High-Granular Calorimeters

Particle showers, as previously described, are complex phenomena that are captured and measured inside calorimeters. In this and some of the next chapters, there will be a description of the process that finally led us to the development of an ML algorithm that allowed a significant step forward in the simulation of showers in very complex and high granular calorimeters. At the beginning of this thesis project, one of the most popular and discussed technologies was the emerging Graph Neural Networks already introduced in Sec. 4.6. However, GNNs suffer the curse of dimensionality, as the number of edges in a graph grows quadratically with the number of nodes. This is a problem in the context of the HGCal (cfr. Sec. 3.4), where the number of cells is approximately 3 million. This would cause an enormous number of edges and a very inefficient adjacency matrix at least memory-wise, making the training of a GNN on the full graph unfeasible.

7.1 Dataset

The data for this work has been simulated using the CMS Offline Software [124] (CMSSW) based on GEANT4 [125]. The simulation has been restricted to the electromagnetic part of the calorimeter (CEE). The chosen geometry is the v14, a prototype of the HGCal [2] with 28 layers divided into hexagonal modules, each composed of two layers of lead absorber and two copper-tungsten layers that act as sensors. The tracker has been removed from the simulated geometry to avoid the effect of interaction with the material before the calorimeter. In addition, the noise in the simulation has been disabled.
As described in Section 3.4, the CEE comprises hexagonal cells of two different sizes. The outer region of the calorimeter is the low-density one where the plates are filled with cells sized $\sim$1.1cm^2. The inner region plates instead, have cells sized $\sim$0.5cm^2 thus having a higher density.
The simulated showers are from photons ranging from 10 to 100 GeV originating from the interaction point. An arbitrary combination of $\eta = 2$ and $\phi = 1.51$ has been fixed and all the photons impact the calorimeter at the same location. This means that the showers also come with an angle as shown on the overlay plot in Fig.7.1. During the GEANT4 simulation, up to 4000 points are stored depending on the energy of the incoming photon, so to reduce cardinality, the energy depositions lying in the same cell are summed up.
For training, 200 000 showers have been simulated, uniformly distributed in the 10 to 100 GeV incident energy range.

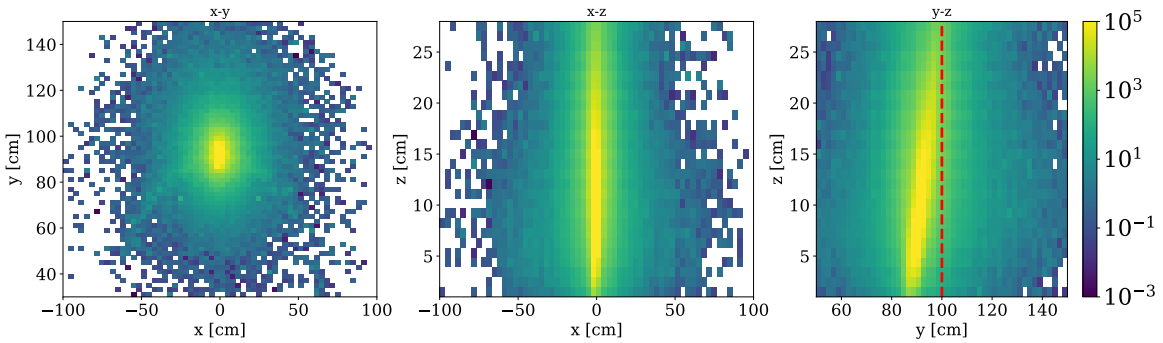


Figure 7.1: Overlay of 20000 GEANT4 generated photon showers. In the y-z plane, the incidence angle is visible.

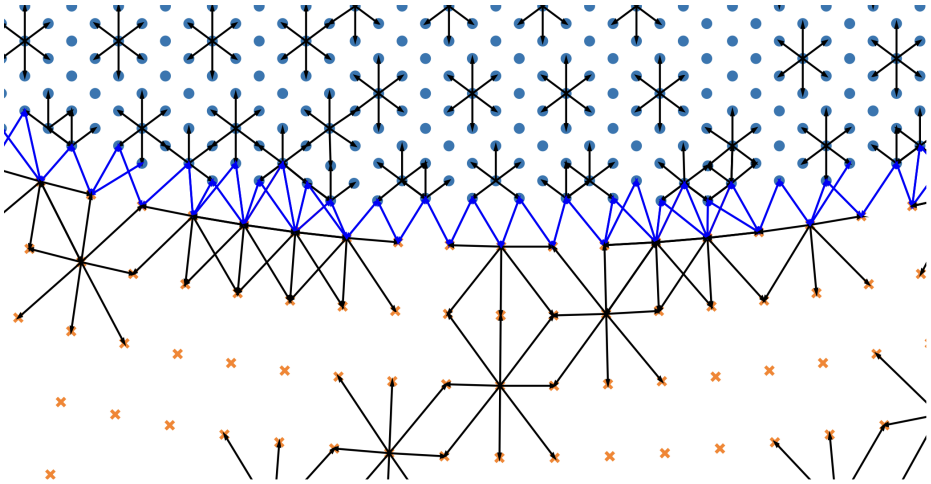


Figure 7.2: Schematization of the HGCal geometry and graph. Form CMMSW we are provided with cell properties like the cell position, material, and the neighbor cells within the same sub-detector (silicon in blue and scintillator in orange) To build edges across different sub-detectors, we applied a rule of maximum distance between cells of 0.5cm. This provides a fully connected graph across the whole calorimeter. Image by Moritz Scham, DESY.

7.2 Dynamic Graph Convolutional Networks

To overcome the issue of the cardinality, we embarked on the development of a novel GNN architecture, called Dynamic Graph Convolutional Network (DGCN), that could be able to handle the high granularity of the HGCal. A straightforward way of doing a GNN model based on showers would be to build graphs out of the particle showers constructing the adjacency matrix using some kind of clustering algorithm, e.g. a kNN. However, this approach would be disregarding one crucial piece of information that is already available, i.e. the geometry of the detector. The latter can be used to build an adjacency matrix that is more meaningful. As mentioned before, the HGCal cells have a hexagonal shape, where each silicon cell is connected to its six nearest neighbors, while the scintillator cells can be connected to their 8 closest neighboring cells. A schematization of the neighbors' connection is displayed in Figure 7.2. The full-geometry graph built on this idea is going to represent the adjacency matrix to be used as a look-up table to dynamically build the subgraphs of the showers.

Once the full-geometry graph has been built, smaller graphs can now be constructed using the hits in the showers as nodes and defining the connection based on the look-up table.

Motivation for a Geometry-based Adjacency Matrix

To justify the use of an adjacency matrix based on the geometry of the detector, we performed a test using graphs built on the same photon showers, but using three different algorithms to construct the edges: kNN clustering, radial clustering, and the geometry-based adjacency matrix. We then trained three identical GNNs with an energy-regression task and compared the performance of the models. The results are shown in Figure 7.3. The geometry-based adjacency matrix clearly outperforms the kNN clustering algorithm and it shows better stability compared to the radial clustering algorithm. This is an indication that the geometry-based adjacency matrix was a meaningful choice for the HGCal.

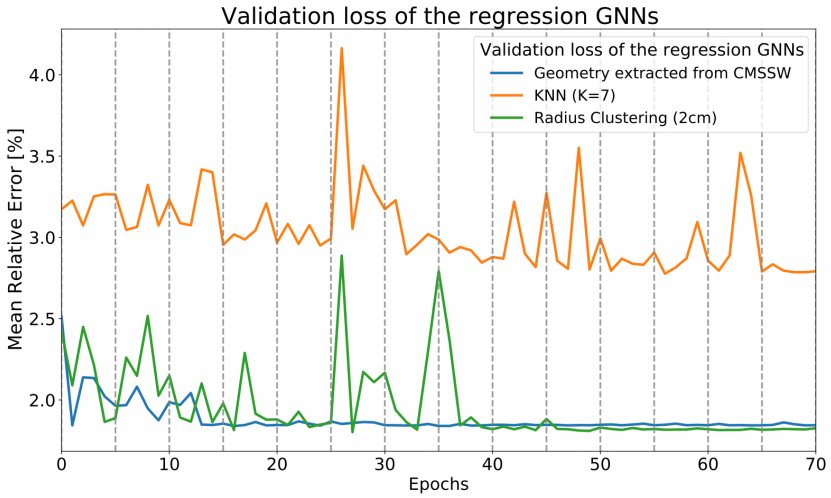


Figure 7.3: Comparison of the performance of GNNs trained on graphs built using three different algorithms. The geometry-based adjacency matrix shows either better performance or more stable one when compared to the other two.

Generation Algorithm

To generate the showers using our model, we decided to approach it in a way that could be as efficient as possible and that would mimic the actual development of a shower. We initially decided to start with a single node that is the seed of the shower and it's initialized with an energy value based on the entire energy that the shower should have taking into consideration the sampling fraction of the calorimeter at hand. Then, we iteratively add nodes to the graph, based on our look-up table (the full graph), and have the generator redistribute said energy to the new nodes. This would grow the graph systematically and we would have a pruning process in place that cuts nodes that don't meet the threshold energy set by the detector resolution. This process is repeated until the energy is fully distributed and the shower is fully developed.

However, this came with a number of setbacks:

- The redistribution of the energy was not effective, as the algorithm was not working on randomly initialized nodes, and it was not taking place in practice.
- Pruning could also end up shattering the continuity of graph as per fluctuations during the redistribution, crucial nodes could be cut off as they were momentarily below the threshold.
- The pruning process was very likely to destroy the energy profile as, taking into consideration the sparse nature of the showers, the number of neighbors to which the energy would be redistributed to would be much higher than the actual number that comprise the final shower causing in fact some energy loss in the process when nodes below the threshold are cut.
- The selected model, a GAN, was not able to learn the distribution of the showers, as during training we would be comparing fully developed shower graphs to the partially developed ones. Masking would have not been an option as developing graphs with few nodes would have a higher energy than the real graph. All this compined together would make the learning process very difficult for the generator and way too easy for the discriminator.
- Finally, there was not a clear way to iteratively generate the showers with a GAN, which is notoriously a one-shot generation model and would need to be completely rethought to accommodate the iterative nature of the idea.

To tackle some of these problems, we updated the idea: instead of having a single GAN generating the whole shower, we would train multiple GANs. Each of the GANs would have as a training set a pooled version of the GEANT4 showers starting with a few nodes up to the full shower, and the generated graph would be initialized with random noise instead of value set a priori. A schematic view of the algorithm is shown in Figure 7.4. This would have simplified the generation procedure and would have introduced a fairer training process. However, the technical complexity of the idea was too high, and with time new technologies emerged, and we decided to switch to a more state-of-the-art approach with point clouds as we will see in the next chapter.

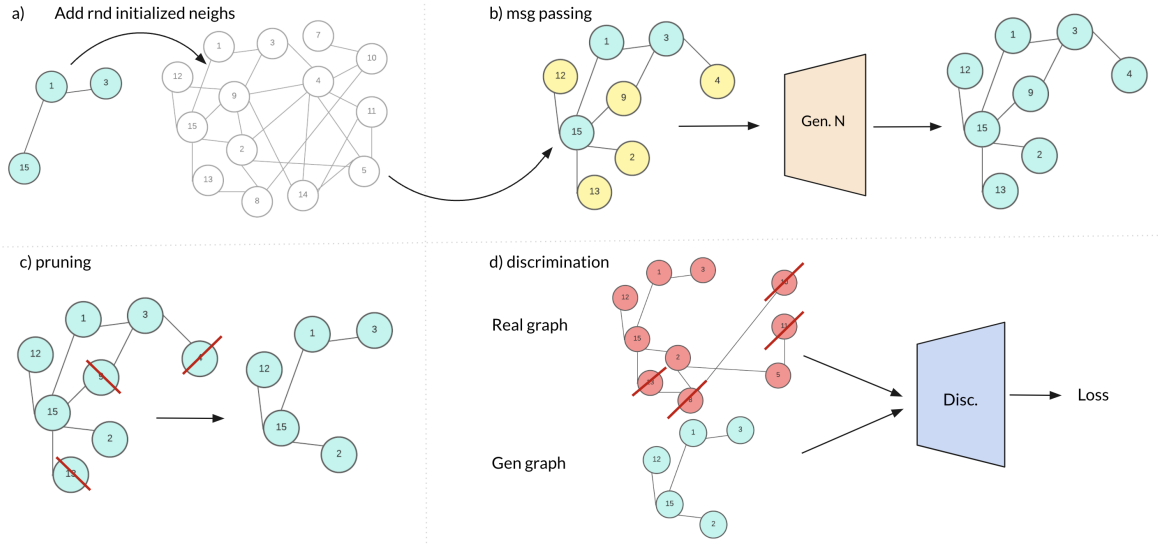


Figure 7.4: Schematic view of the DGCN generation algorithm. The generator is trained on a pooled version of the GEANT4 showers and the generated graph is initialized with random noise.

8 EPiC GAN for 4-Dimensional Calorimeter Data

This chapter will introduce the next step taken in trying to produce an accurate simulation of the CMS HGCal calorimeter. To do so we did an intermediate study on a less complicated dataset that still retains some characteristics of the problem we want to tackle, but loses most of the irregularities that come with the HGCal setup. The idea was to have a proof of concept on this simpler dataset before moving on to the more complex one.

Several approaches have been pursued to approach calorimeter simulation using different models like autoencoders [126–131], GANs [132–143], Flow-based models [144–152], and diffusion models [153–163].

In the past years, GANs have emerged as powerful tools in artificial intelligence, offering a promising avenue for addressing the computational challenges associated with simulating electromagnetic showers. GANs, known for their ability to learn and replicate complex data distributions, provide a unique opportunity to generate realistic point cloud representations of electromagnetic showers with improved efficiency compared to traditional simulation methods.

Here the GAN approach based on EPiC layers is introduced and the techniques employed and the limitations of this approach will be discussed.

8.1 Data Samples

The ILD [164] detector represents one of the two proposed detector concepts for the ILC [165]. It has been designed to cater to Particle Flow [166]. ILD integrates precise tracking and vertexing capabilities with excellent hermiticity and highly granular electromagnetic and hadronic calorimeters. In this study, we focus on showers simulated in the Si-W ECal. This consists of 30 active silicon layers in a tungsten absorber stack. The silicon sensors have a size of 5x5 mm^2. In the original dataset, the showers were projected into a regular grid of 30x30x30 cells in which there was a 1-on-1 correspondence between the geometry and artificial grid cells. To reduce the complexity of the dataset for proof-of-concept, we clustered the geometrical cells into a 10x10x10 grid thus reducing the granularity. In Fig. 8.1 one can see the difference for the same stack of showers in the original granularity vs. the reduced granularity.

The data is then converted to a point cloud format. As the positions of the hits always correspond to the center of the cell in the grid format, we choose to dequantize them by adding noise to randomly shift the hit positions within the cells. Since the cells have a volume of 1x1x1, to ensure the shifted point lays inside its original cell, this is done by simply adding uniform noise $\mathcal{U}(-0.5, 0.5)$ to the coordinates. Finally,

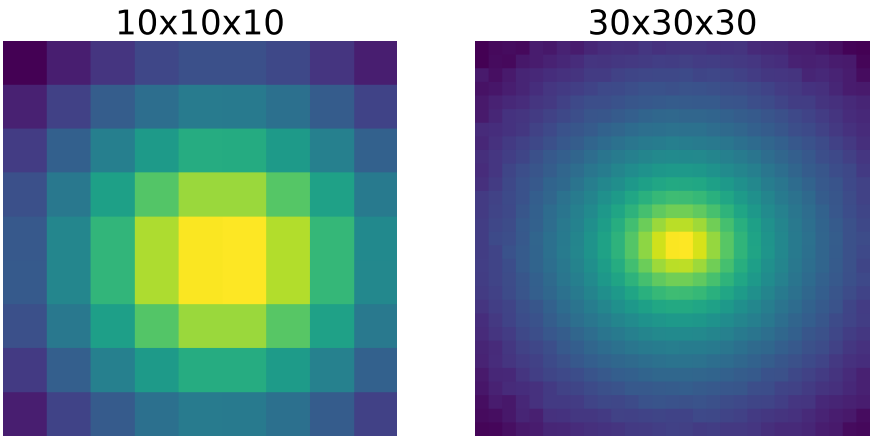


Figure 8.1: View of the x-y plane of a stack of 15k showers in the 10x10x10 vs the 30x30x30 regular grid.

we preprocess the input features to shift them into the [-1; 1] range with the sole exception of the energy that is left untouched.

For the sake of completness, we also performed studies on showers with higher granularities but we couldn't achieve performances on par with what is shown in Sec. 8.3.

8.2 EPiC GAN Architecture

In this section, we introduce the model architecture of our GAN. The models are implemented using PyTorch [167] and the original code for the EPiC GAN can be found on GitHub[1].

The EPiC Layer

The EPiC layer is a layer that has been introduced in Ref. [168]. Let $C = (\mathbf{g}, P)$ be a 2-tuple point cloud that can be considered an edgeless graph. $\mathbf{g}$ represent the global attributes of C and P is the set of points in the cloud. The EPiC layer acts on both $\mathbf{g}$ and P using the following operations:

$$\begin{aligned} \mathbf{g}' &= \phi^g\left(\mathbf{g}, \rho^{p\to g}(P)\right), \\ \mathbf{p}'_i &= \phi^p\left(\mathbf{g}', \mathbf{p}_i\right) \end{aligned}$$

where $\rho^{p\to g}$ is a pooling operation that aggregates the points in the cloud into a global feature vector, and ϕ^g and ϕ^p are two neural networks that act on the global and point features, respectively.

The EPiC layer can be seen as an edgeless graph, allowing for linear complexity in the number of points in the cloud. The global transformation is performed before the local one, ensuring that the global features are available to the local transformation independently of the number of EPiC layers stacked. The aggregation function includes

[1] https://github.com/uhh-pd-ml/EPiC-GAN

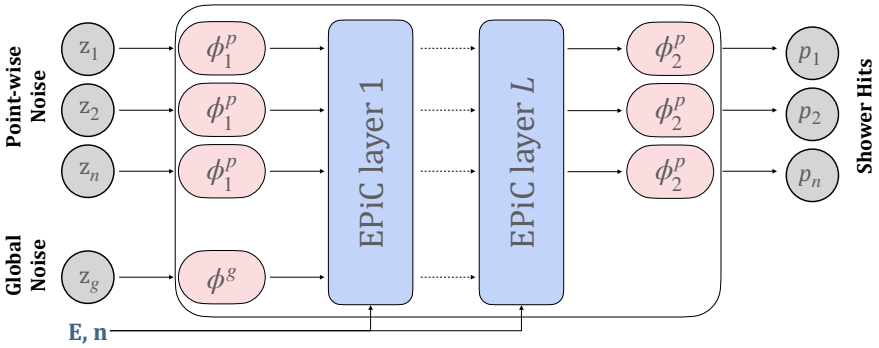
both sum and mean pooling, which empirically showed superior performance with variable-sized clouds.

GAN Architecture

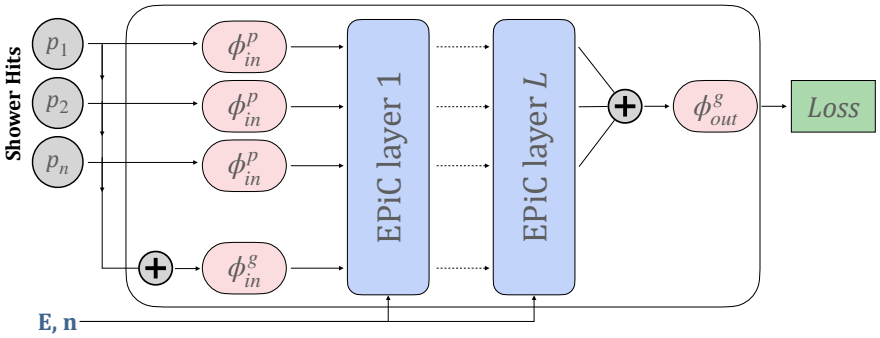
The EPiC GAN is composed of a generator and a discriminator. In the generator, the global noise is first transformed by a two-layer multi-layer perceptron (MLP) into a higher-dimensional global attribute vector, while the point-wise noise transforms a one-layer MLP to create initial point representations. These transformed features are then passed through multiple EPiC layers, which introduce inter-point communication while maintaining permutation equivariance. Each EPiC layer in the generator follows a structured transformation process. First, the global attribute vector is updated based on the aggregated information from the current point features. Then, each point feature is modified based on the updated global attribute vector. The aggregation function, which consists of sum and mean pooling, ensures that relevant information is captured at the global level without explicit pairwise interactions between points. To the original network, we performed some small modifications, i.e. we found that we could achieve high-fidelity generation after five EPiC layer iterations. On top of that, while performing the sum pooling operation we applied a scale factor of 10^{-3} to the result: this is done to account for the high cardinality of the input that led to overly high values in the layers after the sum pooling that where affecting the quality of the training. Finally, we added the energy of the incident photons and the number of shower hits as a conditioning variable to the discriminator and the generator EPiC layers.
The discriminator is composed of an initial multi-layer perceptron to perform feature expansion, followed by three EPiC layers. Sum pooling and mean pooling are also applied here, and the same scale factor is applied to account for the cardinality of the input. In the end, a final MLP is used to perform the binary classification.
During the training, the dataset was divided into batches of a maximum of 128 showers. Contrary to the original version of the EPiC GAN, we don't batch together showers of the same size, but the data is zero-padded and shuffled. The padding is then reduced to the maximum number of hits in the batch during training. Finally, a mask is applied to exclude the remaining zero padding from the loss computation. At the generation step, we apply the same mask as the input data to generate showers of equal cardinality. This ensures that the discriminator compares clouds of the same size.

Multiple GAN training regimes have been tested, e.g. WGAN, LSGAN, etc. Still, the one that yielded the best results is the vanilla GAN training objective in which the discriminator D and the generator G play a minimax game as described in Section 5.1.

Finally, the best epoch is selected by comparing the mean Wasserstein-1 distance of the normalized coordinates, visible energy, and center of gravity distributions. We compare the distribution by generating 2000 showers at fixed energy points, namely 20, 50, 70, and 90 GeV.



(a) Generator



(b) Discriminator

Figure 8.2: Illustration of the EPiC G4N architecture. The discriminator has an extra pooling step to compute the global features of the input showers.

8.3 Results

Physics Performance and Evaluation Scores

Here are now presented the results of the model by giving an overview of its performance measuring the KL divergence [169] on the observables. Although the Wasserstein-1 distance was used to select the best epoch for its better interpretability during training, we prefer the KL divergence for the final score evaluation score since we find it better suited for well-overlapping distributions as the Wasserstein distance would average out under- and overshooting of the distributions, while the KL divergence evaluates them on a bin-by-bin basis. To compute each of the scores we generated 50 000 showers in the uniform energy range [10; 100] GeV divided into 5 batches and we computed the distances to the Geant4 showers for each batch. Finally, we compute the mean and standard deviation of such measurements to get the results shown in

Tab 8.1.

The $\mathrm{KLD}_{E_{vis}}$ measures the distance between the real and the generated visible energy distribution in the whole 10 to 100 GeV energy range, and the $\mathrm{KLD}_{\mathbf{z}}$ is the distance between the hits distributions in the longitudinal direction, the $\mathrm{KLD}_{cog_{\mathbf{z}}}$ is the distance of the center of gravity in said direction, and finally $\mathrm{KLD}_{E_{rad}}$ is the distance between the two radial energy distributions.

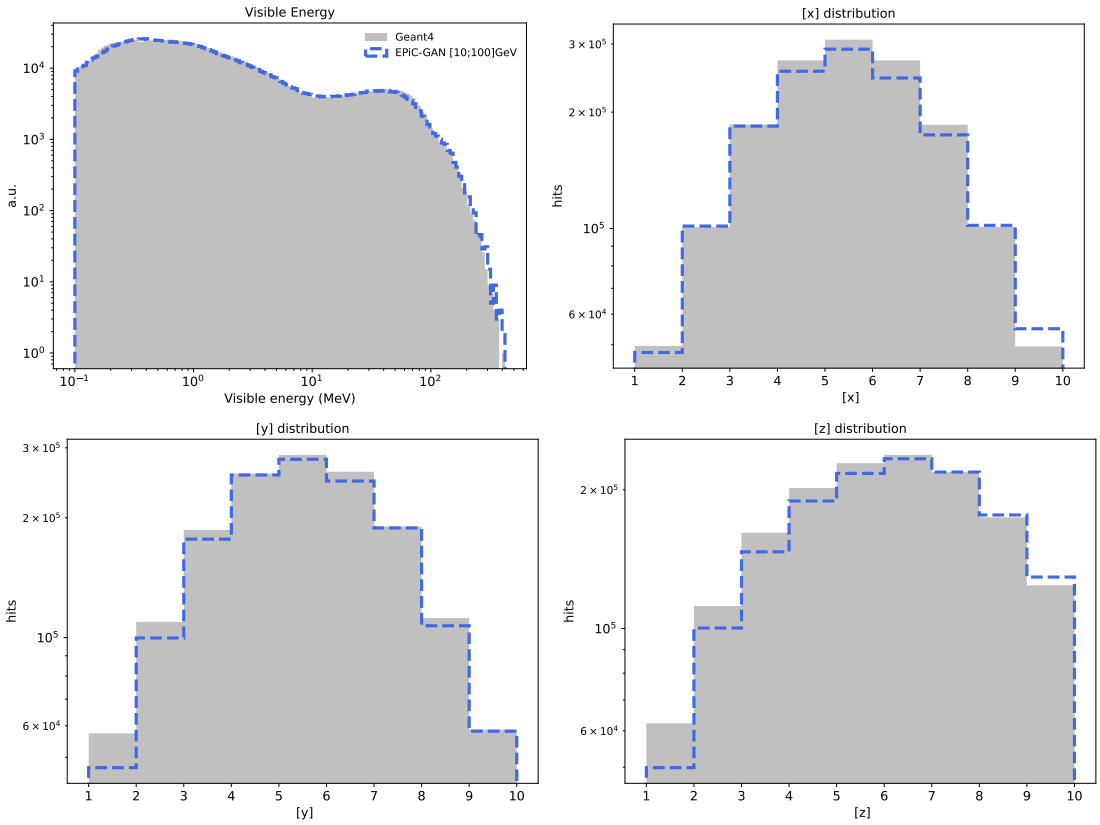


Figure 8.3: Comparison for 10 000 Geant4 photon showers from the test set and 10 000 EPiC GAN generated showers for the point-wise observables. The point cloud has been projected to the 10x10x10 grid after being generated. On top of that, a cut on the energy at 0.1 MeV has been applied.

$\mathrm{KLD}_{E_{vis}}(\times 10^{-4})$	$\mathrm{KLD}_{z}(\times 10^{-4})$	$\mathrm{KLD}_{E_{rad}}(\times 10^{-4})$	$\mathrm{KLD}_{cog_{z}}(\times 10^{-4})$
7.68 ± 0.67	14.94 ± 2.94	4.21 ± 0.68	49.29 ± 1.33

Table 8.1: KLD distances comparison

As can also be seen in Fig.8.3 and **??**, the model generates showers with good fidelity.

Results on Single Energy Shower

We now show the results on showers produced by photons with fixed energy. We selected photons with an energy of 20, 50, 70, and 90 GeV. What can be seen from the single energy plot distribution shown in Fig. 8.5 is that the EPiC-GAN performs better as the energy of the incoming photon increases. This can be solved with targeted training on the more problematic energy ranges.

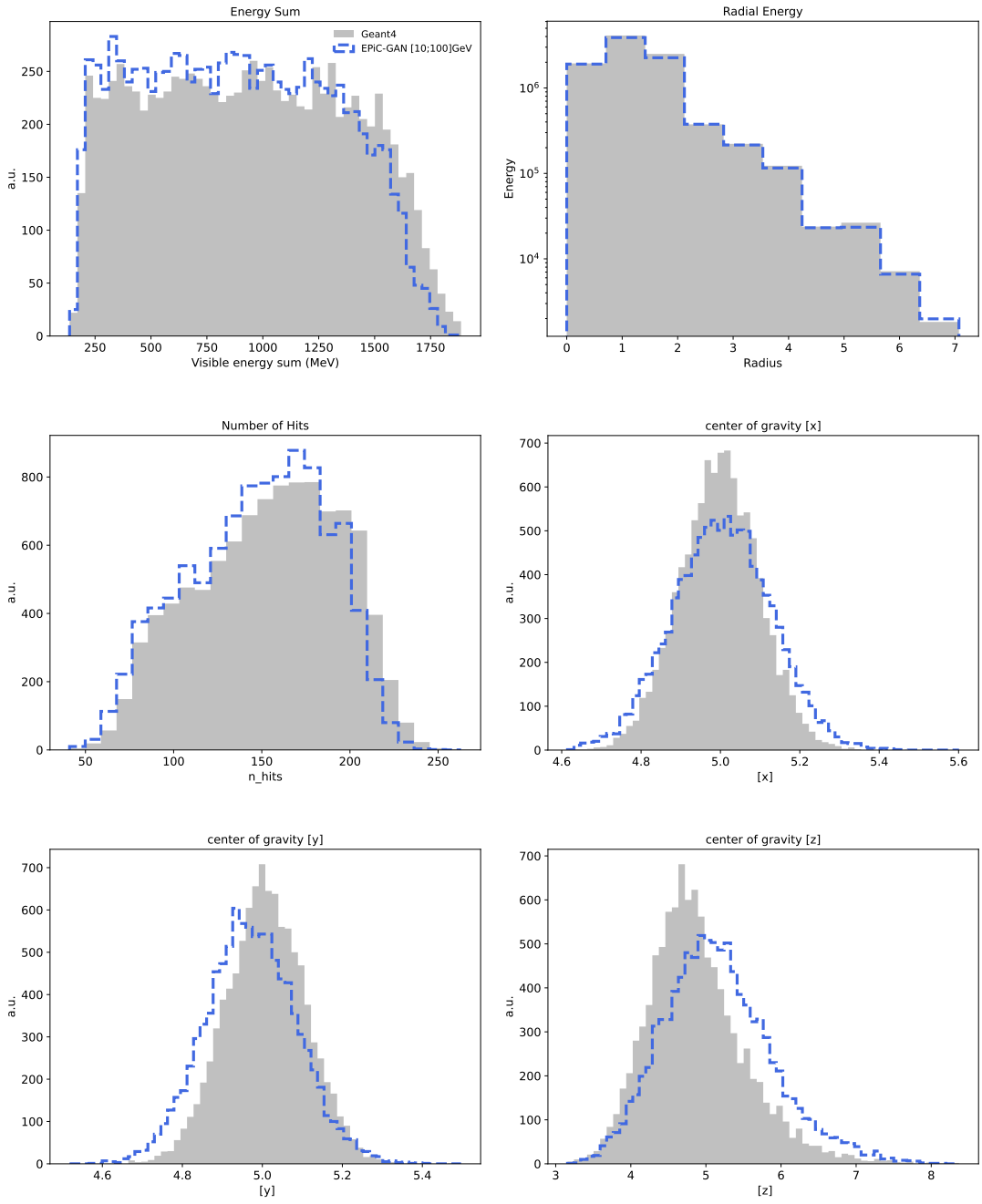


Figure 8.4: The shower observables are compared for the same 10000 photon showers.

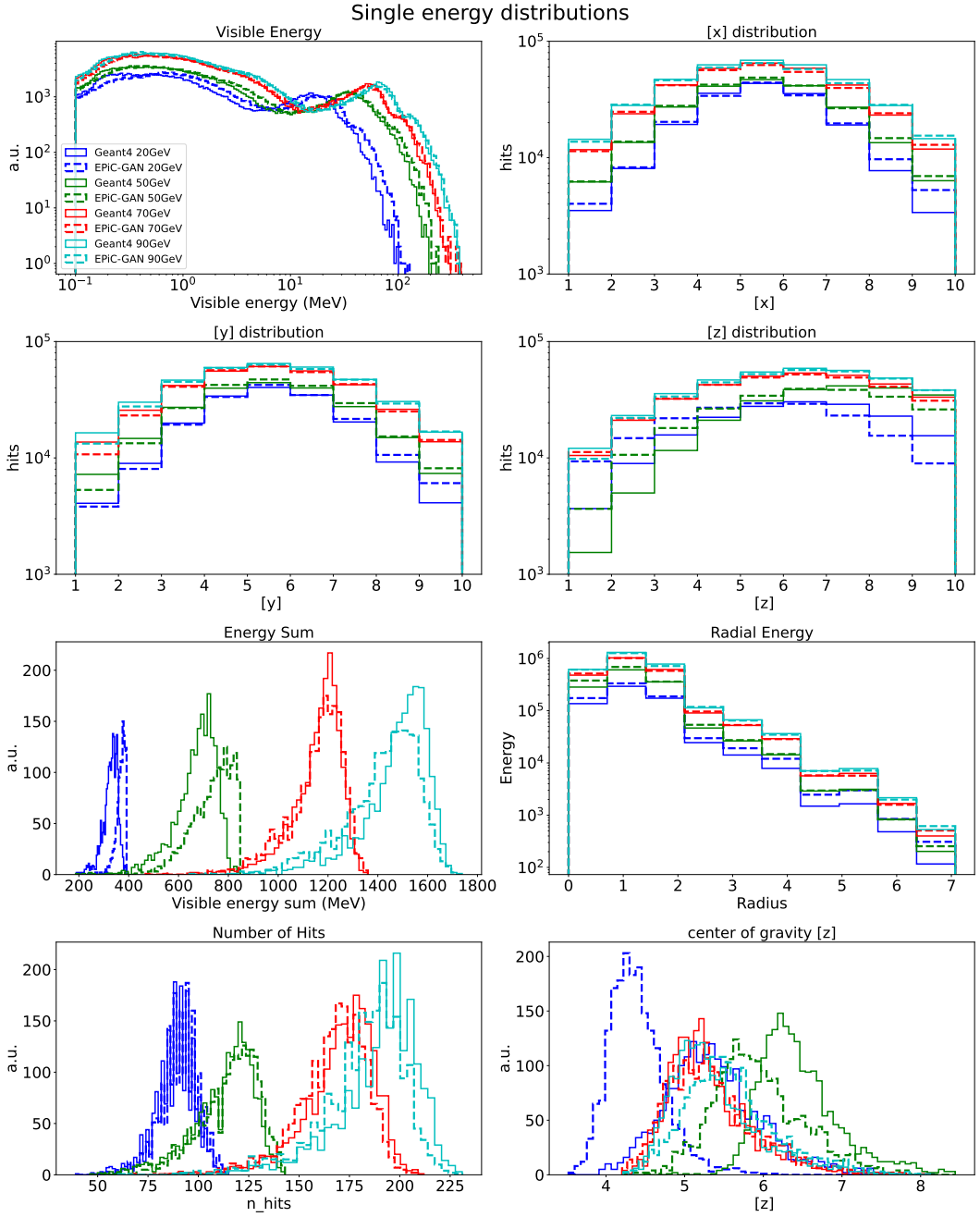


Figure 8.5: Distribution plots of the single energy photon showers. At low energy, one can see that the model has a harder time reproducing the distributions. This can be addressed with targeted training in the problematic energy range, instead of training on the the full range of the energy spectrum.

E_i(GeV)	CPU t(ms)/shower	GPU t(ms)/2k showers	Avg. N-hits
10-100	6.02 ± 0.36	2.95±0.04	236 ±64
20	4.67±0.04	2.95±0.05	140±13
50	6.10±0.10	2.97±0.07	196±21
70	8.60± 0.06	3.00±0.07	276±25
90	9.82± 0.04	3.14±0.09	308±29

Table 8.2: Time comparison for different energy slices on an AMD EPYC 7543 CPU and an NVIDIA® A100 GPU with 80GB of memory. In all cases, 2000 showers were generated with the shown energies. The presented values are means and standard deviations over 10 runs.

Timing results

Finally, in Tab. 8.2 we provide an overview of the generation times. We produced 10 times a set of 2000 showers with both a uniform energy distribution ranging from 10 to 100 GeV and fixed energy values on a single CPU and an NVIDIA® A100 GPU. The mean and standard deviation of the measurements are presented. The timing results are well aligned with the general idea of GAN architectures being extremely high speed generators. As the tipical GEANT4 generation speed lays in the realm of seconds-per-event, we see the GAN performing a factor 1000 better than that. However, fair comparison cannot be performed as we have no data on a lower granularity GEANT4 setup matching the one showed in this study.

8.4 Conclusions

The EPiC-GAN showed interesting results. It was able to tackle a dataset with more than double the cardinality of the previous study on jets that comprises data points with up to 150 constituents. A higher complexity is also introduced by the extra feature of the dataset, with not only spatial coordinates but also the energy of the showers. This is shown to be modeled in good fashion in the results, both statistically, as per the visible energy, and in its distribution as per the center of energy distributions. The model also shows good performance in the single energy distribution, with the only exception of the lowest energy range. This can be addressed with targeted training on the problematic energy range. In terms of timing, GANs are known to be very fast models and with a generation time of 2.95 ms per 2000 showers on an NVIDIA® A100 GPU this is confirmed in this study.
Even with all these great improvements, unfortunately, the EPiC-GAN showed instability when trained on even higher cardinality datasets. After trying several different variations of the model, we couldn't achieve a good level of performance on more complex and granular datasets. This limitation of the model brought us to the conclusion that the EPiC-GAN is not the best model to tackle the HGCal simulation problem and we moved on to a different model concept, diffusion models, that will be presented in the next chapter.

9 CaloClouds for the CMS HGCal

Since the proposal by [133], has investigated more and more the use of generative surrogate models to speed particle physics simulation. These surrogates mimic the so-called simulation chain usually employed in particle physics, with specific models targeting the hard process [170–176], parton shower and hadronization [177–189], and the interaction of particles with complex detectors [190].

In this domain of calorimeter simulation, a key frontier is the development of models that can handle data with the complexity and spatial resolution expected of calorimeters like the CMS High Granularity Calorimeter (CMS HGCal) [2] or calorimeters proposed for future colliders [164]. Two important breakthroughs in this regard were i) adopting point-cloud-based models [159, 160, 185, 191, 192] that only simulate actual hits in the detector (instead of explicitly modeling also the empty cells) and ii) the switch to powerful generative models based on flows and diffusion [144–151, 153–159, 161–163]. One such model that achieved state-of-the-art performance in the simulation of the electromagnetic calorimeter for the planned International Large Detector (ILD) at the International Linear Collider (ILC) is CaloClouds II [160]. In this work, we show how CaloClouds II can be adapted to model showers in the CMS HGCal.

The CMS HGCal is being developed for the future High-Luminosity upgrade at the LHC [1] and will replace the current end-cap calorimeters. It is designed to have a high granularity to accurately reconstruct the particle's energy. It includes both an electromagnetic and a hadronic part and utilizes a hybrid approach that includes both silicon sensors and scintillator materials, to optimize over the range of energies.

The calorimeter has a quite complex structure (cfr. Section 3.4), where the layers feature a hexagonal tiling pattern. This allows for the efficient packing of detector modules, enhancing spatial resolution and minimizing dead areas between the detectors. In particular, the cells have variable sizes depending on their location: cells closer to the beam pipe are smaller for better resolution, while those in the peripheral area, are larger to balance the need for coverage. In total, one end-cap of the calorimeter will consist of over 3 million cells. These features combined, make this detector a hard challenge for fast generation.

Finally, the CMS HGCal also provides timing information per hit, which is crucial in reconstructing and separating piled-up events. The HGCal calorimeter aims to reach a time resolution of $\sim$30 picoseconds. Beyond investigating the performance of CaloClouds II in the task of simulating HGCal data, we will also consider the inclusion of time (in addition to energy) as a generative target. The rest of this chapter is organized as follows: in Sec. 9.1 a description of the dataset and how it was produced can be found. In Sec. 9.2 we will introduce the generative model used for this piece of work, while Sec. 9.3 will describe the selected geometry and our projection algorithm. Finally, in Sec. 9.4 we show the achieved results, followed by our conclusions in Sec. 9.5.

9.1 Data: format and processing

The simulation chain for a calorimeter comprehends different steps:

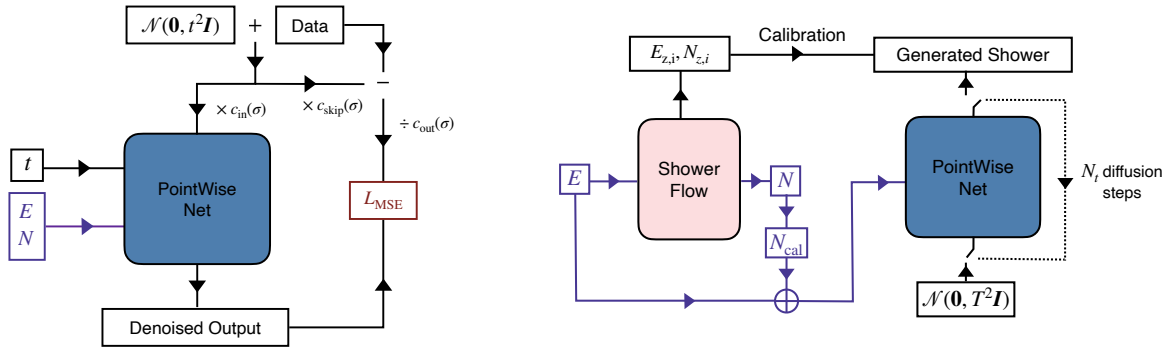
- SimHits step: the interaction of the particles is simulated by the Monte Carlo and saved as if the detector had infinite time and energy resolution. The interaction information is stored in the center of the cells of the geometry, allowing for multiple hits to be in the same spot with different energy;
- Digis step: includes the pile-up and the response from the front-end electronics;
- RecHits: pre-processed digits are used to reconstruct the hits in the cells with energy and time information;

The data used to train the model is the output of the SimHits step as it is the most raw and unprocessed data available and it is in a format that suits very well the model's input. This implies a higher cardinality (up to ∼4000) of lower energy hits. However, the CaloClouds II models [159, 160] can better reproduce physical distributions by generating points at a higher granularity that can then be clustered into voxels rather than trying to place the final clustered hit exactly in the correct cell.
The chosen format for the sets is that of point clouds as it is the preferred one for CaloClouds II [160] as it is more efficient since it does not require the full geometry of the detector to be stored wasting memory on empty cells. The simulated features are the x, y in the layers' plane (in centimeters), the layers z in the depth direction (from 1 to 28), the *energy* of the SimHit (in MeV), and the *time-of-hit* of the particle (in nanoseconds). This makes up for a total of a 5-dimensional vector that is used as an input to the model.
On top of the input features, the number of particles of the shower and the incident energy of the photon are also used in the model's training as conditional features.
The data undergoes some preprocessing steps before being fed to the network. The coordinates of the hits as well as the time and the conditioning features are normalized into the [-1, 1] range. On top of that, the dataset is all zero-padded and ordered by the number of points. This last step is done so that during training the padding can be cut to the maximum number of SimHits in the current batch to ensure that there is a minimal amount of padding in the training.

For training, we generated 180 000 showers, and for testing and performing a classifier test, we have 200 000 showers split into 20 000 fir metrics evaluation and the rest for the classifier test. Both datasets are uniformly distributed in the 10 to 100 GeV incident energy range. On top of that, two sets of 20 000 showers each have been simulated for fixed energy points of 20 and 70 GeV to study the model's performance.

9.2 Generative Model

The model used in this work is the CaloClouds II described in [160]. As the model is based on Ref. [160], a brief explanation of its components follows. First, we have the *Shower Flow*, a normalizing flow [193] model trained to generate conditioning features and used to calibrate the output of the diffusion model [194]. More specifically, the *Shower Flow* generates the number of points and the energy of the incoming photon as conditioning features for the sampling step. On top of that, to calibrate the output



(a) Training (b) Sampling

Figure 9.1: Illustration of the training and sampling procedure of the CaloClouds II model.
(a) The model is trained at a random continuous time step t with a condition on the showers' energy E and its number of points N. The L_{MSE} is approximated by a mean squared error (MSE) between the noised data and the denoised output. The scaling functions c_{in}, c_{out}, and c_{skip} are defined following [195].
(b) During sampling the Shower Flow conditioned on E generates the number of SimHits for the showers as well as their observables for calibration. Then the *PointWise Net* iteratively denoises the noise to produce the showers. Finally, the center of energy in the x direction is calibrated as well as the energy and the number of points per layer.

distributions, it generates the total visible energy, the energy and number of points per layer, and the center of energy (CoE) in X. As for the diffusion model, we use a continuous-step diffusion paradigm, usually referred to as *k-diffusion* based on [195]. The actual diffusion model is a *PointWise Net* composed of six ConcatSquash Linear layers [196] to which the five input features are fed plus the conditioning features as per Ref. [160]. We can see a diagram of the training-sampling procedure in Figure 9.1.

Training and Sampling

The model in CaloClouds II was trained for 3M iterations with a batch size of 128 using the Adam optimizer [62] with a fixed learning rate of 10^{-4}. As the final model, we use an exponential moving average (EMA) of the model weights. The final number of function evaluations (NFE), i.e. the number of ODE solver steps to be performed, has been chosen after comparing different values and has been chosen to be 50, as it is the one that gives the best quality/speed ratio.

9.3 Projection to Geometry

To compare the output distribution of CaloClouds II to the synthetic data, we must first project the point cloud to the actual geometry, as in the chosen step of the Geant4 simulation, the SimHits are centered in the cell they belong to. To do so, we

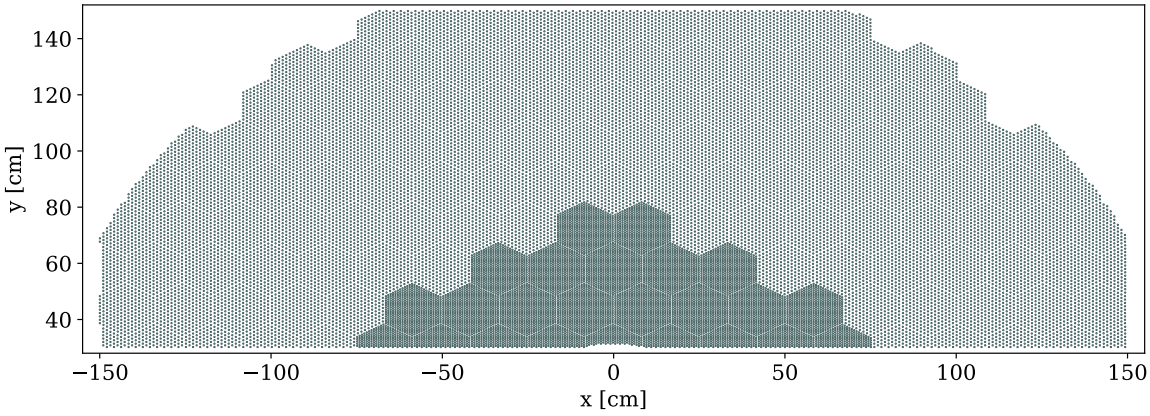


Figure 9.2: In this figure we see the 28th layer of the Electromagnetic part of the HGCal after the selected geometry cut. The outer area is the lower-density region (sparser dots) while closer to the beam-pipe we have the higher-density region with smaller hexagonal cells.

first have to define which cut of the geometry we want to consider and then develop an algorithm that allows such a procedure.

Geometry Selection

First, we extract the geometry from a ROOT [197] file that contains the coordinates of every cell in the CEE. This allows us to see the structure of the calorimeter and exclude regions that are scarsely or not populated as the showers will not reach them or the geometry presents structural gaps, such as the beam pipe. In Figure 9.2 one can see which section of the calorimeter has been selected. After the cut, we are left with circa 800 000 cells out of 3M, divided into ∼27 000 cells per layer.

Projection Algorithm

Projecting the point cloud to the geometry can become very complicated, given the architecture of the calorimeter, made of different-sized hexagons, that are not always perfectly aligned neither vertically, horizontally, or longitudinally, and are sometimes reshaped to match the borders of the plate they reside in. We decided to tackle this by using a simple approach: for each hit, we compute its distance to every cell center in the same layer and then assign the closest center coordinates to the current SimHit. In return, this approach is quite slow, but optimizing such projection is outside the scope of this research work. The pseudo-code for this algorithm reads as follows:

9.4 Results

We show the model's results by comparing some low-level and high-level distributions to the GEANT4 ground truth. We then provide a score for such observables using the

```
for every_shower do
    for layer in CEE do
        closest_cell ← min(points − centers);
        points ← closest_cell;
    end
end
```

Algorithm 1: Algorithm to find the closest cell centers to each point in the generated showers. After finding them, it updates the coordinates of the points to the centers' coordinates.

Wasserstein distance [63] and a classifier test. Finally, we benchmark the generation time of the model against GEANT4.

Physics Distributions

In this Section, we confront several observables to the GEANT4 test set. We compare showers generated in the uniformly distributed energy range from 10 to 100 GeV and two test sets of fixed energy, at 20 GeV and 70 GeV. All the presented distributions have been projected with the introduced Algorithm 1.
In Figure 9.3, we show four relevant quantities and the ratio of the generated showers to the GEANT4 test set dataset. The presented distributions are of the energy of the SimHits, the radial energy distribution, the per-shower energy sum, and the longitudinal distribution across the layers. These plots show the quality of the CALOCLOUDS II model with the ratio to GEANT4 being consistently close to 1. To add more shower level information, we present in Figure 9.4 the CoE for the three directions. Here again, the ratio plots demonstrate the performance of the generator. The 2-dimensional energy-weighted overlay can be seen in Figure 9.5 where it is visible how the model is capable of correctly reproducing the showers' angle and features.
We plot the time coordinate generated with CALOCLOUDS II. We show two representations, one of the overall time-of-hit (TOH) of the SimHit, which again measures up to the standards of the previously shown plots, and the TOH distribution per layer.
Finally, we show the energy sum plot and number of hits for the two single energy points in Figure 9.7.

Evaluation Scores

To assess the quality of the distributions with some solid numbers and not only by comparing histograms, we picked some high-level features of the showers. The features produced consist of the number of hits (N) in the showers, the sampling fraction defined as the ratio between the incident photon energy and the total visible energy in the calorimeter (E_{samp}), the center of energy in the three spatial coordinates ($CoE_{[x,y,z]}$), the radial energy distribution (E_{rad}), the energy per layer defined as the energy sum per layer (E_l), and the mean TOH per layer (μ_t).
To get comparable values of these distributions, we computed the 1-Wasserstein distance (W_1) between GEANT4 and CALOCLOUDS II distribution pairs. Following com-

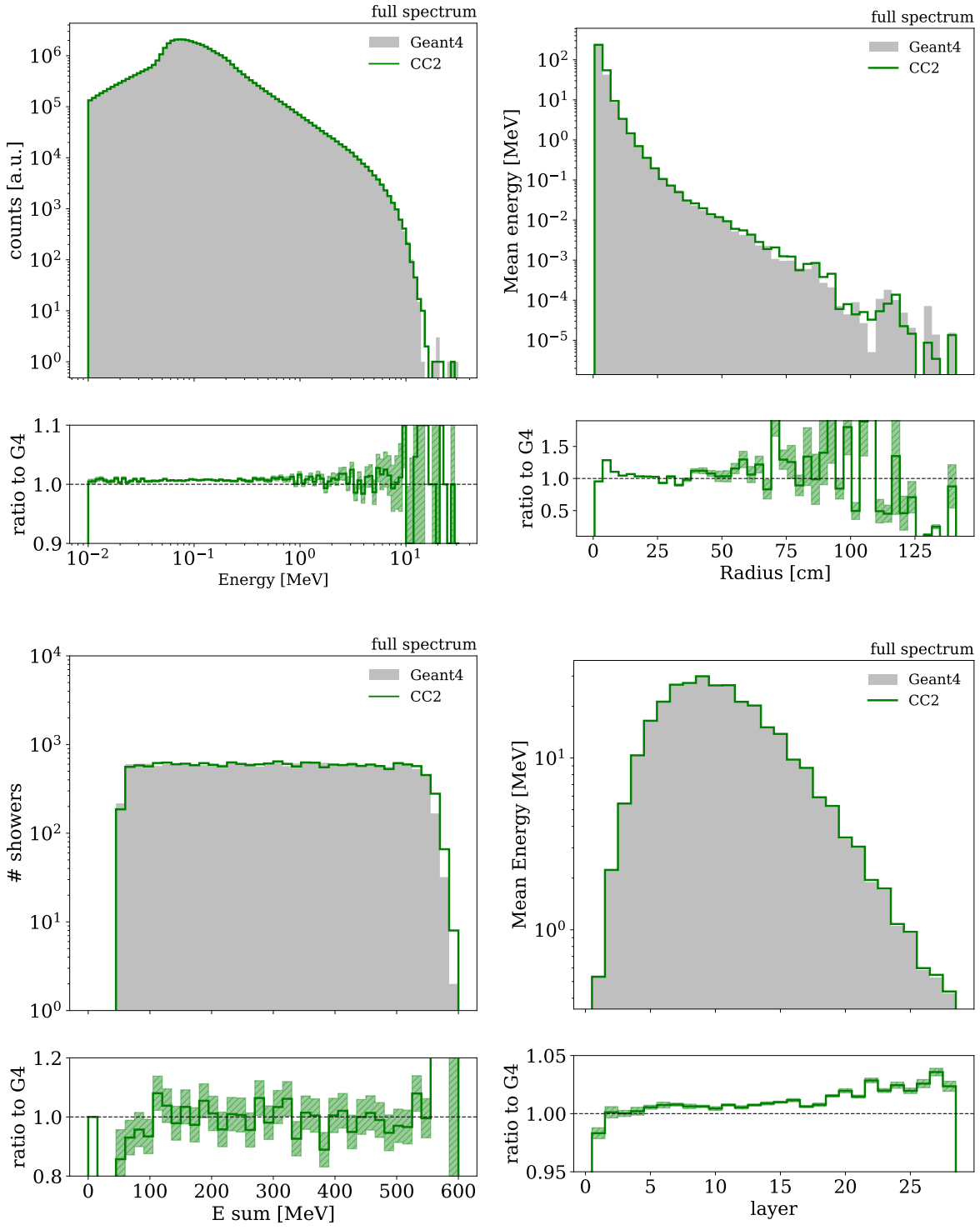


Figure 9.3: From top left to bottom right, we present the SimHit energy distribution, the radial energy distribution, the energy sum, and the mean energy-per-layer distribution. From the ratio plot, one can see the high fidelity of the CaloClouds II generated showers.

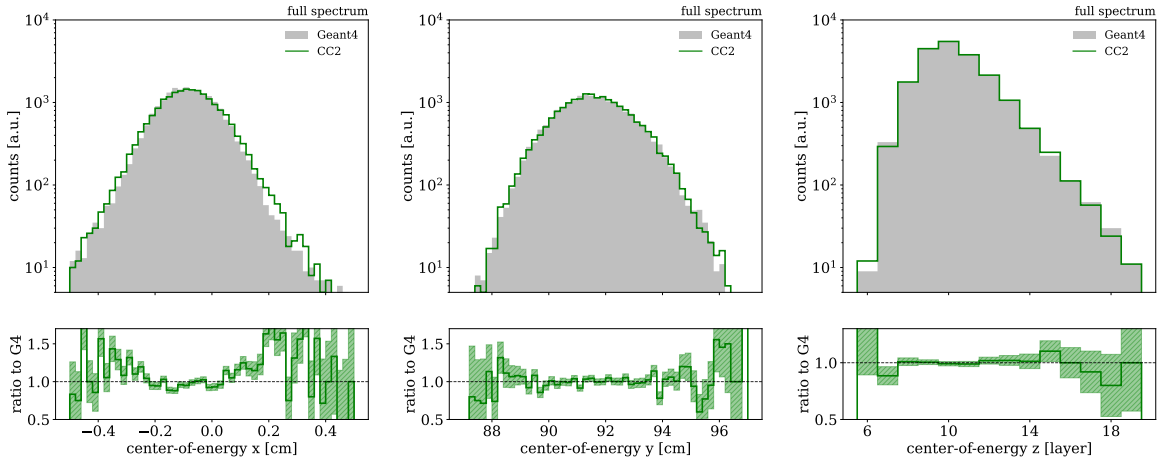


Figure 9.4: Center-of-energy distributions computed along the three axes. In the x direction, the model produces a slightly too narrow distribution and is then calibrated using the *Shower Flow* while preserving the SimHits distribution quality.

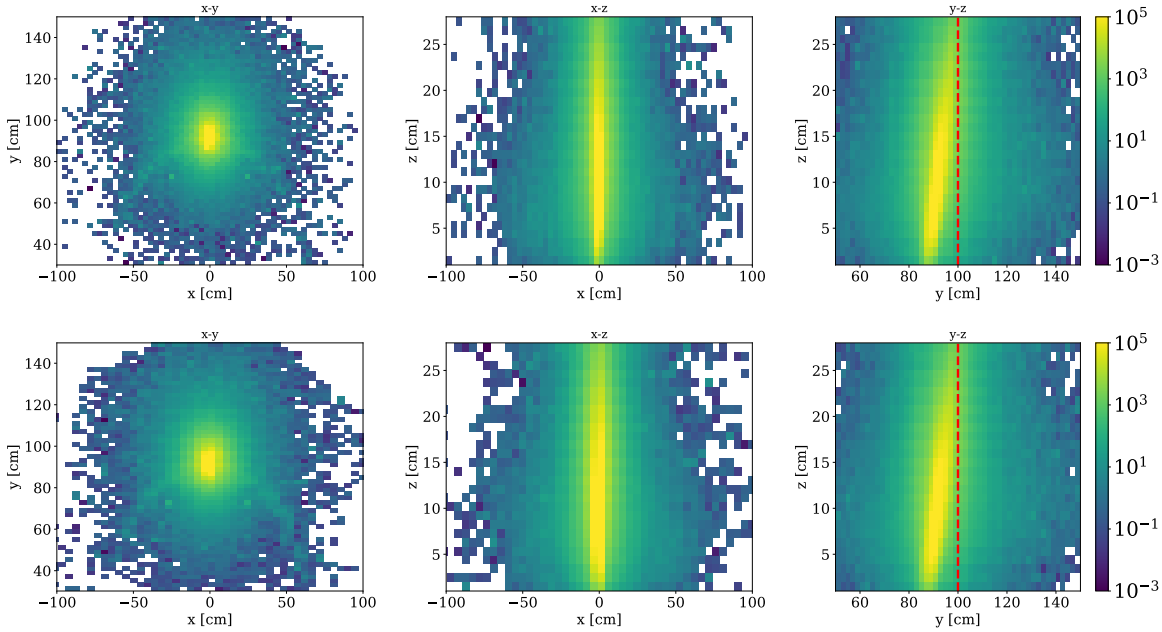


Figure 9.5: Overlay of 20000 showers. The first row represents the GEANT4 dataset, while the bottom row is the CALOCLOUDS II simulated showers.

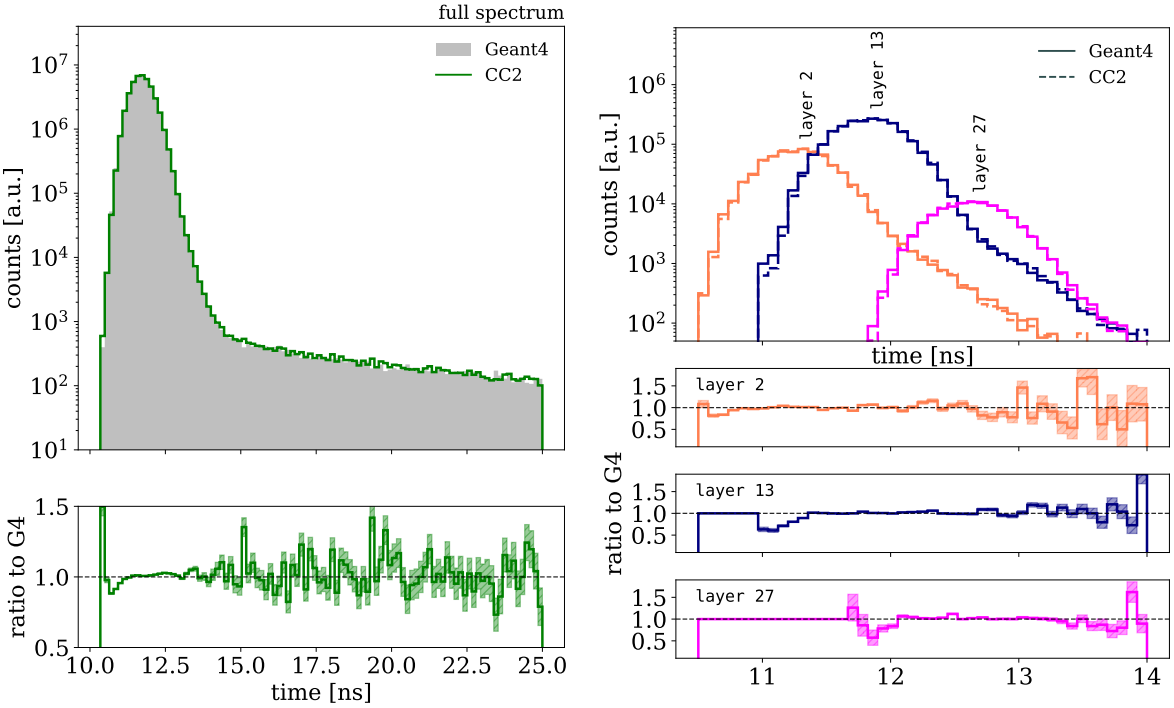


Figure 9.6: On the left we see the overall time-of-hit distribution, while on the right we present the single-layer time distributions. In terms of statistics, the plots show high fidelity to the GEANT4 showers. What these plots cannot capture is the time correlation from layer to layer which is still not perfect as some particles may result as faster than light.

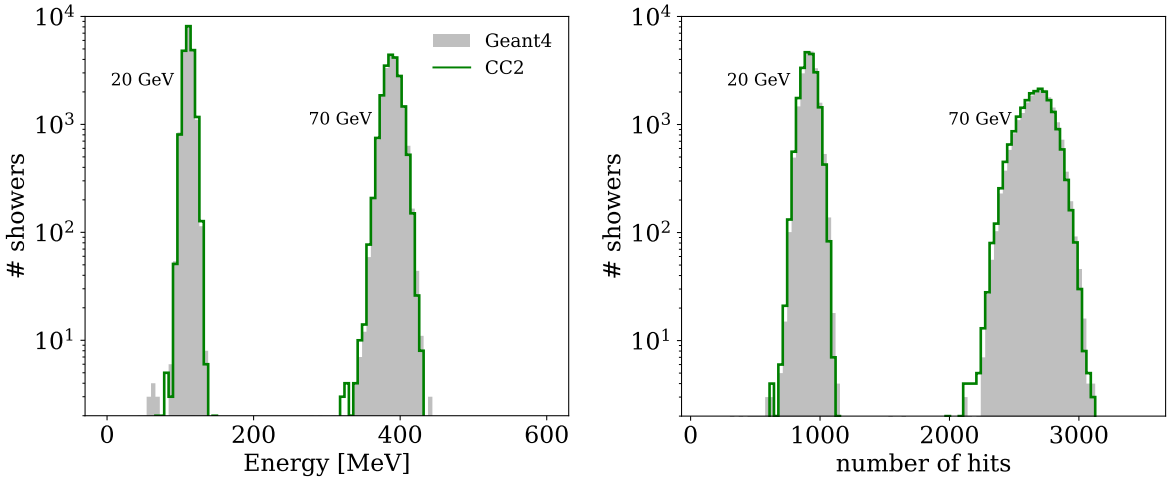


Figure 9.7: On the left side we can see the energy sum distributions for the two fixed incident energy points: 20 GeV and 70 GeV. We find the number-of-hits distributions for the same two fixed energy points on the right side. Again we observe a good match between the GEANT4 baseline and the CALOCLOUDS II generated distributions.

mon practices with W_1 distances evaluations, we compare 20000 GEANT4 showers to 20000 CALOCLOUDS II showers and repeat this operation 10 times with uniformly distributed independent showers. We then report the mean and standard deviation of such measurements. To better understand the scores' goodness, we also report the values obtained for the W_1 for GEANT4 vs. GEANT4 distributions.

	W_1^N	$W_1^{E_{samp}}$	$W_1^{\mu_t}$	$W_1^{E_l}$	$W_1^{E_{rad}}$	W_1^{CoEx}	W_1^{CoEy}	W_1^{CoEz}
Truth	2.2 ± 0.6	2.0 ± 0.5	1.2 ± 2.9	1.6 ± 3.1	6.2 ± 7.1	1.9 ± 0.3	1.3 ± 0.2	1.3 ± 0.4
CC2	17.5 ± 0.8	3.1 ± 0.4	9.6 ± 4.6	11.4 ± 4.3	11.3 ± 7.8	11.2 ± 0.6	6.4 ± 0.7	5.8 ± 0.3

Table 9.1: Wasserstein scores for several quantities. The values are calculated on 10 independent sets of 20000 uniformly distributed showers each. We also do the same calculation using the two GEANT4 sets, to have the truth values for such scores.

Classifier Scores

To give a more comprehensive score to the showers instead of their marginal distributions, we performed two Classifier tests: one including high-level features based on all 5 low-level features and one excluding the time coordinate. Such classifier tests are becoming more widely used as an evaluation method for generative models and similar tests can be found in Refs [130, 145, 146, 148, 151, 184, 198]. The tests were made using high-level features of the showers, namely the number of SimHits overall and per layer, the energy per layer, the center of energy in all three directions, the sampling fraction, and the mean and standard deviation per layer of the TOH. The dataset comprises 200000 points per feature for both GEANT4 and CALOCLOUDS II

showers and an 80-10-10% split is applied.
The classifier is a fully connected neural network with three layers (containing 32, 16, and 8 nodes) with LeakyReLU [64] activation functions and one output node with Sigmoid activation. Using a binary cross-entropy loss, the trainining is performed with the Adam optimizer [62] for 10 epochs for each dataset. The final model epoch is chosen based on the lowest validation loss.
The test has been repeated 10 times and we report the mean AUC and its standard deviation in Table 9.2.

AUC w/o time features	AUC all features
0.7738 ± 0.0016	0.9884 ± 0.0008

Table 9.2: From the table we can see the performance of the CaloClouds II model without and with the time coordinate. Without the time features, the AUC score results in an improvement of 23% over Ref. [160], although the different setups make it an unfair comparison. Unsurprisingly, adding time to the features causes the AUC to increase as the model can more easily distinguish the generated shower.

The results show a better performance when exluding the time-related features, as the model is not designed to capture correlations between points and some particles may result faster than light. A further study of this phenomenon and an attempted improvement of the performance is been conducted in a related work of a bachelor student supervised also by me [199]

Timing

We now benchmark the average time needed to generate the showers using the CaloClouds II model on a GPU and CPU. The timing results are presented in Tab. 9.3.

Hardware	Simulator	NFE	Batch Size	Time / Shower [ms]	Speed-up (w/o proj)
CPU	Geant4			3153.63± 1490.68	×1
	CaloClouds II	50	1	714.33 ± 288.22	×4
GPU	CaloClouds II	50	64	31.04 ±0.67	×102

Table 9.3: The simulation speed performance of CaloClouds II compared to the baseline Geant4 simulator on a single core of an AMD® EPYC 7543 32-Core (CPU) and on an NVIDIA® A100 with ∼40 GB of memory (GPU). We generated 2000 showers with incident energy distributed uniformly between 10 and 100 GeV. The values presented are the mean and standard deviations for 10 runs. The NFE column refers to the number of function evaluations (model passes) performed to diffuse the showers.

From the table, one can see that the speed-up offered by CaloClouds II is not excessively big. This is not a surprise as *k-diffusion* models are intrinsically slow, even

if they can be optimized by tuning the NFEs. That being said, we already demonstrated in Ref. [160] how such a model can be distilled into a faster generator, thus drastically increasing the simulation speed without compromising the fidelity.

9.5 Conclusions

The idea of developing a generative model to tackle the challenge of simulating complex and fine-grained detectors like the HGCal led to the design of CaloClouds II. The proposed model could simulate point-cloud-based events in a cell geometry-independent fashion. Following up on that piece of work, we now show not only how this model is suited for geometries with such a high number of geometry cells, but also how we can push its capabilities further by teaching it to generate the time-of-hit coordinate, a feature in which correlation is important to model correctly. A first for as far as we know to the day.
In conclusion, the CaloClouds II model is capable of electromagnetic shower simulation with high cardinality maintaining high fidelity. We compared several observables to the Geant4 simulation, the baseline for such tasks and the results show high performance. The time coordinate, which is a key feature to reproduce in light of the future upgrade of the LHC, is statistically very well modeled but can be improved in the context of correlation. The two classifier tests confirmed that the time coordinate was still not perfect, but the overall fidelity of the shower was converging to a good grade.

10 Conclusion

The field of high-energy physics (HEP) is driven by a fundamental quest to understand the nature of the universe at its most elementary level. Particle physics experiments, particularly those conducted at large collider facilities such as the Large Hadron Collider (LHC) at CERN, have provided profound insights into the fundamental constituents of matter and the forces governing their interactions. These experiments rely on highly sophisticated detector systems designed to capture and analyze the vast amounts of data generated by high-energy particle collisions. One of the critical components of these detectors is the calorimeter, an instrument used to measure the energy of particles through their interactions with matter. Calorimetry plays an essential role in reconstructing the properties of fundamental particles and identifying signatures of new physics beyond the Standard Model.

This dissertation has explored the landscape of machine learning techniques applied to high-granular calorimeter simulation in high-energy physics research. The work has been driven by the need to address the computational challenges associated with traditional Monte Carlo simulation methods, which become increasingly demanding for complex detector systems like the CMS HGCal. The investigation has ventured into various machine learning approaches, including generative adversarial networks (GANs), normalizing flows, and diffusion models, to generate realistic calorimeter showers with significantly reduced computational costs compared to traditional methods. These models have demonstrated remarkable success in approximating the complex distributions of particle interactions within calorimeters, enabling efficient and high-fidelity simulations.

Chapter 6 showed a unified a graph-based architecture that could achive high-level perfromance on diverse datasets using shared hyperparameters. Chapter 7 of the dissertation introduces the idea of Dynamic Graph Neural Networks (DGCNs) for simulating high-granular calorimeters. DGCNs address the computational challenges posed by the high granularity of the CMS HGCal by dynamically constructing subgraphs of particle showers based on the detector geometry. This approach could have allowed for efficient processing of calorimeter data while preserving the relational structure of particle interactions. The motivation behind using GNNs stems from the highly granular nature of modern calorimeters, where particle showers can be represented as graph structures for more flexible and efficient data processing. GNNs leverage the relational structure of calorimeter hits to learn spatial correlations and improve event reconstruction. DGCNs concept would have been an architecture that dynamically builds subgraphs of particle showers based on the detector geometry. This approach makes use of the actual geometrical information to build more meaningful connections between the nodes of the shower which has been shown to yield and advantage compared to classical clustering algorithms. The model starts with a seed node and iteratively adds new nodes based on the detector geometry and the energy distribution of the shower. A pruning process is employed to remove nodes that do not meet

the energy threshold, ensuring efficiency and maintaining the shower's energy profile. This approach had the potential to improve the accuracy and scalability of calorimeter simulations, but the technical challenges and the fast-paced advancements in the generative machine learning field led us to switch to more advanced, scalable, faster, and efficient paradigms.

Chapter 8 presents the EPiC GAN, a generative model designed for calorimeter data, focusing on accurately capturing the spatial distribution and energy deposition of showers. The EPiC GAN was evaluated on a simpler dataset that shares characteristics with the CMS HGCal but lacks its irregularities and granularity. Results demonstrate the model's ability to generate realistic calorimeter showers with good performance and efficiency.

The EPiC GAN builds upon the success of GANs in generating synthetic data that closely resemble real detector responses. GANs consist of two neural networks - a generator and a discriminator - trained in a competitive framework. The generator learns to produce realistic samples, while the discriminator attempts to distinguish between real and generated data. Through this adversarial process, the generator improves its ability to generate high-quality calorimeter showers that mimic those produced by Monte Carlo simulations.

The EPiC GAN incorporates an architecture that leverages spatial and energy information to generate calorimeter data. The model employs EPiC layers, which are edgeless graph layers that allow for linear complexity in the number of points in the cloud. This architecture enables efficient processing of high-dimensional calorimeter data while preserving good scalability with increasing cardinality. The EPiC GAN was trained on a dataset of photon showers with varying energy levels, and its performance was evaluated using metrics such as the Wasserstein distance and KL divergence to compare the generated showers to the ground truth data. The results show that the EPiC GAN can accurately reproduce the spatial distribution and energy deposition of showers, demonstrating its potential for simulating complex calorimeter systems with improved accuracy and scalability compared to traditional methods. For as good as the results were, this model still lacked in performance when the showers were approaching higher granularities, making it a non-optimal choice for a calorimeter like the HGCal.

Because of that, Chapter 9 focuses on the CaloClouds II model. It leverages the concept of diffusion models, which simulate the gradual transformation of noise into structured calorimeter showers through a series of stochastic steps. This approach offers high-quality sample generation with strong theoretical guarantees, making it well-suited for applications in high-energy physics.

The CaloClouds II model employs a combination of normalizing flows and the continuous time diffusion model and the data employed are not reconstructed showers but raw Geant4 SimHit showers. This implies a higher cardinality as Geant4 hits are not clustered into the geometrical cell of the calorimeter. This puts fewer constraints on the model, allowing small imperfections in the point cloud output to be mitigated. The generated showers are then projected to the physical geometry in a postprocessing step. This approach ensures that the generated showers are consistent

with the physical constraints of the detector, and allows for a fair comparison to the GEANT4 data. A key feature of this study is the incorporation of the timing information. As previously mentioned, this is an important addition to the CMS HGCal and as such it requires high fidelity when reproduced with our model. In Chapter9.4, we saw an accurate statistical generation of this feature that can still be improved with a higher focus on the point-wise correlations.

The adaptation of CALOCLOUDS II to the CMS HGCal has demonstrated its ability to handle complex geometries and high-granularity data. The model's ability to generate realistic calorimeter showers while incorporating timing information represents a significant advancement in calorimeter simulation techniques. The CALOCLOUDS II model was trained on a dataset of photon showers in the CMS HGCal, and its performance was evaluated by comparing the generated showers to the ground truth data using various metrics, including the Wasserstein distance and a classifier test.

In conclusion, this dissertation has explored the application of machine learning techniques to high-granular calorimeter simulation in high-energy physics research. The study has investigated various machine learning approaches, including GNNs, GANs, normalizing flows, and diffusion models, to generate realistic calorimeter showers with significantly reduced computational costs compared to traditional methods. The development and application of specific models, such as DGCNs, EPiC GAN, and CALOCLOUDS II, have demonstrated the potential of machine learning to transform calorimeter simulation and data analysis, paving the way for more accurate, efficient, and scalable simulation frameworks in future high-energy physics experiments.

Bibliography

[1] I. Zurbano Fernandez et al. "High-Luminosity Large Hadron Collider (HL-LHC): Technical design report". In: 10/2020 (Dec. 2020). Ed. by I. Béjar Alonso et al. DOI: 10.23731/CYRM-2020-0010.

[2] The CMS collaboration. *The Phase-2 Upgrade of the CMS Endcap Calorimeter*. Tech. rep. Geneva: CERN, 2017. DOI: 10.17181/CERN.IV8M.1JY2. URL: https://cds.cern.ch/record/2293646.

[3] CMS Offline Software and Computing. *CMS Phase-2 Computing Model: Update Document*. Tech. rep. Geneva: CERN, 2022. URL: https://cds.cern.ch/record/2815292.

[4] Steven Weinberg. "A Model of Leptons". In: *Phys. Rev. Lett.* 19 (1967), pp. 1264–1266. DOI: 10.1103/PhysRevLett.19.1264.

[5] S. L. Glashow. "Partial Symmetries of Weak Interactions". In: *Nucl. Phys.* 22 (1961), pp. 579–588. DOI: 10.1016/0029-5582(61)90469-2.

[6] David J. Gross and Frank Wilczek. "Ultraviolet Behavior of Nonabelian Gauge Theories". In: *Phys. Rev. Lett.* 30 (1973). Ed. by J. C. Taylor, pp. 1343–1346. DOI: 10.1103/PhysRevLett.30.1343.

[7] H. David Politzer. "Reliable Perturbative Results for Strong Interactions?" In: *Phys. Rev. Lett.* 30 (1973). Ed. by J. C. Taylor, pp. 1346–1349. DOI: 10.1103/PhysRevLett.30.1346.

[8] Antonio Pich. "The Standard Model of Electroweak Interactions". In: *2010 European School of High Energy Physics*. Jan. 2012, pp. 1–50. arXiv: 1201.0537 [hep-ph].

[9] MissMJ, Cush. *Standard Model of Elementary Particles*. [Online; accessed 10-September-2022]. 2019. URL: https://upload.wikimedia.org/wikipedia/commons/0/00/Standard_Model_of_Elementary_Particles.svg.

[10] Mark Thomson. *Modern particle physics*. New York: Cambridge University Press, 2013. ISBN: 978-1-107-03426-6.

[11] The LHCb collaboration. "Observation of the resonant character of the $Z(4430)^-$ state". In: *Phys. Rev. Lett.* 112.22 (2014), p. 222002. DOI: 10.1103/PhysRevLett.112.222002. arXiv: 1404.1903 [hep-ex].

[12] The LHCb collaboration. "Observation of $J/\psi p$ Resonances Consistent with Pentaquark States in $\Lambda_b^0 \to J/\psi K^- p$ Decays". In: *Phys. Rev. Lett.* 115 (2015), p. 072001. DOI: 10.1103/PhysRevLett.115.072001. arXiv: 1507.03414 [hep-ex].

[13] F. Englert and R. Brout. "Broken Symmetry and the Mass of Gauge Vector Mesons". In: *Phys. Rev. Lett.* 13 (1964). Ed. by J. C. Taylor, p. 321. DOI: 10.1103/PhysRevLett.13.321.

[14] Peter W. Higgs. "Broken symmetries, massless particles and gauge fields". In: *Phys. Lett.* 12 (1964), p. 132. DOI: 10.1016/0031-9163(64)91136-9.

[15] Peter W. Higgs. "Broken Symmetries and the Masses of Gauge Bosons". In: *Phys. Rev. Lett.* 13 (1964). Ed. by J. C. Taylor, p. 508. DOI: 10.1103/PhysRevLett.13.508.

[16] The ATLAS collaboration. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Phys. Lett. B* 716 (2012), p. 1. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214 [hep-ex].

[17] The CMS collaboration. "Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC". In: *Phys. Lett. B* 716 (2012), p. 30. DOI: 10.1016/j.physletb.2012.08.021. arXiv: 1207.7235 [hep-ex].

[18] Albert Einstein. "The foundation of the general theory of relativity." In: *Annalen Phys.* 49.7 (1916). Ed. by Jong-Ping Hsu and D. Fine, pp. 769–822. DOI: 10.1002/andp.19163540702.

[19] K. Becker, M. Becker, and J. H. Schwarz. *String theory and M-theory: A modern introduction*. Cambridge University Press, Dec. 2006. ISBN: 978-0-511-25486-4, 978-0-521-86069-7, 978-0-511-81608-6. DOI: 10.1017/CBO9780511816086.

[20] Carlo Rovelli. *Quantum gravity*. Cambridge Monographs on Mathematical Physics. Cambridge, UK: Univ. Pr., 2004. DOI: 10.1017/CBO9780511755804.

[21] Y. Fukuda et al. "Evidence for Oscillation of Atmospheric Neutrinos". In: *Phys. Rev. Lett.* 81 (8 Aug. 1998), pp. 1562–1567. DOI: 10.1103/PhysRevLett.81.1562. URL: https://link.aps.org/doi/10.1103/PhysRevLett.81.1562.

[22] Q. R. Ahmad et al. "Direct Evidence for Neutrino Flavor Transformation from Neutral-Current Interactions in the Sudbury Neutrino Observatory". In: *Phys. Rev. Lett.* 89 (1 June 2002), p. 011301. DOI: 10.1103/PhysRevLett.89.011301. URL: https://link.aps.org/doi/10.1103/PhysRevLett.89.011301.

[23] M. Aker et al. "Direct neutrino-mass measurement based on 259 days of KATRIN data". In: (June 2024). arXiv: 2406.13516 [nucl-ex].

[24] Nicola Cabibbo. "Unitary Symmetry and Leptonic Decays". In: *Phys. Rev. Lett.* 10 (12 June 1963), pp. 531–533. DOI: 10.1103/PhysRevLett.10.531. URL: https://link.aps.org/doi/10.1103/PhysRevLett.10.531.

[25] Makoto Kobayashi and Toshihide Maskawa. "CP-Violation in the Renormalizable Theory of Weak Interaction". In: *Progress of Theoretical Physics* 49.2 (Feb. 1973), pp. 652–657. ISSN: 0033-068X. DOI: 10.1143/PTP.49.652. eprint: https://academic.oup.com/ptp/article-pdf/49/2/652/5257692/49-2-652.pdf. URL: https://doi.org/10.1143/PTP.49.652.

[26] J. H. Christenson et al. "Evidence for the 2π Decay of the K_2^0 Meson". In: *Phys. Rev. Lett.* 13 (4 July 1964), pp. 138–140. DOI: 10.1103/PhysRevLett.13.138. URL: https://link.aps.org/doi/10.1103/PhysRevLett.13.138.

[27] G.'t Hooft. "Naturalness, Chiral Symmetry, and Spontaneous Chiral Symmetry Breaking". In: *Recent Developments in Gauge Theories*. Ed. by G.'t Hooft et al. Boston, MA: Springer US, 1980, pp. 135–157. ISBN: 978-1-4684-7571-5. DOI: 10.1007/978-1-4684-7571-5_9. URL: https://doi.org/10.1007/978-1-4684-7571-5_9.

[28] Manuel Drees. "An Introduction to supersymmetry". In: *Inauguration Conference of the Asia Pacific Center for Theoretical Physics (APCTP)*. Nov. 1996. arXiv: hep-ph/9611409.

[29] Mark Thomson. *Modern Particle Physics*. Cambridge University Press, 2013.

[30] Lyndon Evans and Philip Bryant. "LHC Machine". In: *JINST* 3 (2008), S08001. DOI: 10.1088/1748-0221/3/08/S08001.

[31] The ATLAS Collaboration et al. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08003. DOI: 10.1088/1748-0221/3/08/S08003. URL: https://dx.doi.org/10.1088/1748-0221/3/08/S08003.

[32] The CMS Collaboration et al. "The CMS experiment at the CERN LHC". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08004. DOI: 10.1088/1748-0221/3/08/S08004. URL: https://dx.doi.org/10.1088/1748-0221/3/08/S08004.

[33] The LHCb Collaboration et al. "The LHCb Detector at the LHC". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08005. DOI: 10.1088/1748-0221/3/08/S08005. URL: https://dx.doi.org/10.1088/1748-0221/3/08/S08005.

[34] The ALICE Collaboration et al. "The ALICE experiment at the CERN LHC". In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08002. DOI: 10.1088/1748-0221/3/08/S08002. URL: https://dx.doi.org/10.1088/1748-0221/3/08/S08002.

[35] The CMS collaboration. "The CMS Experiment at the CERN LHC". In: *JINST* 3 (2008), S08004. DOI: 10.1088/1748-0221/3/08/S08004.

[36] "CMS, the Compact Muon Solenoid: Technical proposal". In: (Dec. 1994).

[37] Tai Sakuma. "Cutaway diagrams of CMS detector". In: (2019). URL: https://cds.cern.ch/record/2665537.

[38] The CMS collaboration. "The CMS trigger system". In: *JINST* 12.01 (2017), P01020. DOI: 10.1088/1748-0221/12/01/P01020. arXiv: 1609.02366 [physics.ins-det].

[39] W. Adam et al. "The CMS Phase-1 Pixel Detector Upgrade". In: *JINST* 16.02 (2021), P02027. DOI: 10.1088/1748-0221/16/02/P02027. arXiv: 2012.14304. URL: https://cds.cern.ch/record/2748381.

[40] Paolo Azzurri. "The CMS Silicon Strip Tracker". In: *Journal of Physics: Conference Series* 41.1 (May 2006), p. 127. DOI: 10.1088/1742-6596/41/1/011. URL: https://dx.doi.org/10.1088/1742-6596/41/1/011.

[41] "The CMS electromagnetic calorimeter project: Technical Design Report". In: (1997).

[42] "The CMS hadron calorimeter project: Technical Design Report". In: (1997).

[43] Francesca Cavallari. "Performance of calorimeters at the LHC". In: *Journal of Physics: Conference Series* 293 (2011), p. 012001. URL: https://api.semanticscholar.org/CorpusID:121783334.

[44] "The CMS muon project: Technical Design Report". In: (1997).

[45] Albert M Sirunyan et al. “Performance of the reconstruction and identification of high-momentum muons in proton-proton collisions at $\sqrt{s} = 13$ TeV”. In: *JINST* 15.02 (2020), P02027. DOI: 10.1088/1748-0221/15/02/P02027. arXiv: 1912.03516 [physics.ins-det].

[46] H. Bethe. “Zur Theorie des Durchgangs schneller Korpuskularstrahlen durch Materie”. In: *Annalen der Physik* 397.3 (1930), pp. 325–400. DOI: https://doi.org/10.1002/andp.19303970303. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.19303970303. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19303970303.

[47] William R. Leo. *Techniques for Nuclear and Particle Physics Experiments*. 1987.

[48] S. Navas et al. “Review of particle physics”. In: *Phys. Rev. D* 110.3 (2024), p. 030001. DOI: 10.1103/PhysRevD.110.030001.

[49] Lucio Cerrito. *Radiation and Detectors*. 2017.

[50] Shuji Miyamoto and Ken Horikawa. “New SUBARU Gamma-ray Beam Source and Application Research”. In: *The Review of Laser Engineering* 36 (Jan. 2008), pp. 798–805. DOI: 10.2184/lsj.36.798.

[51] Michele Livan and Richard Wigmans. “Shower Development”. In: *Calorimetry for Collider Physics, an Introduction*. Cham: Springer International Publishing, 2019, pp. 53–73. ISBN: 978-3-030-23653-3. DOI: 10.1007/978-3-030-23653-3_3. URL: https://doi.org/10.1007/978-3-030-23653-3_3.

[52] Shubham Pandey. “Performance of High Granularity Calorimeter prototypes for the CMS HL-LHC upgrade in beam test experiments at CERN”. IISER, Pune, 2022. URL: https://cds.cern.ch/record/2810369.

[53] J. Allison et al. “Recent developments in Geant4”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 835 (2016), pp. 186–225. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2016.06.125. URL: https://www.sciencedirect.com/science/article/pii/S0168900216306957.

[54] J. Allison et al. “Geant4 developments and applications”. In: *IEEE Transactions on Nuclear Science* 53.1 (2006), pp. 270–278. DOI: 10.1109/TNS.2006.869826.

[55] S. Agostinelli et al. “Geant4—a simulation toolkit”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)01368-8. URL: https://www.sciencedirect.com/science/article/pii/S0168900203013688.

[56] J. de Favereau et al. “DELPHES 3: a modular framework for fast simulation of a generic collider experiment”. In: *Journal of High Energy Physics* 2014.2 (Feb. 2014). ISSN: 1029-8479. DOI: 10.1007/jhep02(2014)057. URL: http://dx.doi.org/10.1007/JHEP02(2014)057.

[57] Guenter Grindhammer and S. Peters. “The Parameterized simulation of electromagnetic showers in homogeneous and sampling calorimeters”. In: *International Conference on Monte Carlo Simulation in High-Energy and Nuclear Physics - MC 93*. Feb. 1993. DOI: 10.48550/arXiv.hep-ex/0001020. arXiv: hep-ex/0001020.

[58] Guenter Grindhammer, M. Rudowicz, and S. Peters. "The Fast Simulation of Electromagnetic and Hadronic Showers". In: *Nucl. Instrum. Meth. A* 290 (1990), p. 469. DOI: 10.1016/0168-9002(90)90566-0.

[59] E. Barberio et al. "Fast simulation of electromagnetic showers in the ATLAS calorimeter: Frozen showers". In: *J. Phys. Conf. Ser.* 160 (2009). Ed. by M. Fraternali, Gabriella Gaudio, and Michele Livan, p. 012082. DOI: 10.1088/1742-6596/160/1/012082.

[60] Sasha Glazov. "Fast simulation of showers in the H1 SpaCal calorimeter". In: *J. Phys. Conf. Ser.* 293 (2011). Ed. by Yifang Wang, p. 012024. DOI: 10.1088/1742-6596/293/1/012024.

[61] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG]. URL: https://arxiv.org/abs/1609.04747.

[62] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. DOI: 10.48550/arXiv.1412.6980. eprint: 1412.6980.

[63] Cédric Villani. "The Wasserstein distances". In: *Optimal Transport: Old and New*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, p. 93. ISBN: 9783540710509. DOI: 10.1007/978-3-540-71050-9_6.

[64] Abien Fred Agarap. "Deep Learning using Rectified Linear Units (ReLU)". In: (2018). DOI: 10.48550/arxiv.1803.08375. eprint: 1803.08375.

[65] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[66] Jason Brownlee. *How Do Convolutional Layers Work in Deep Learning Neural Networks?* https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/. 2020.

[67] Maxime Labonne. *Graph Convolutional Networks: Introduction to GNNs*. https://towardsdatascience.com/. 2023.

[68] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[69] Ian Goodfellow et al. "Generative Adversarial Networks". In: *Commun. ACM* 63.11 (Oct. 2020), p. 139. ISSN: 0001-0782. DOI: 10.1145/3422622. eprint: 1406.2661.

[70] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: (Nov. 2014). arXiv: 1411.1784 [cs.LG].

[71] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[72] Ishaan Gulrajani et al. "Improved training of wasserstein gans". In: *Advances in neural information processing systems* 30 (2017).

[73] Xudong Mao et al. “Least squares generative adversarial networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2794–2802.

[74] Lilian Weng. *Flow-based Deep Generative Models*. Ed. by https://lilianweng.github.io/. 2018.

[75] Ricky T. Q. Chen et al. “Neural Ordinary Differential Equations”. In: (June 2018). arXiv: 1806.07366 [cs.LG].

[76] Yaron Lipman et al. “Flow matching for generative modeling”. In: *arXiv preprint arXiv:2210.02747* (2022).

[77] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: (Mar. 2015). arXiv: 1503.03585 [cs.LG].

[78] Arash Vahdat and Karsten Kreis. *Improving Diffusion Models as an Alternative To GANs, Part 1*. Ed. by nvidia developer. 2022.

[79] Aapo Hyvärinen. “Estimation of Non-Normalized Statistical Models by Score Matching”. In: *Journal of Machine Learning Research* 6.24 (2005), pp. 695–709. URL: http://jmlr.org/papers/v6/hyvarinen05a.html.

[80] Yann LeCun and Corinna Cortes. “MNIST handwritten digit database”. In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[81] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: (2009).
Accessed 1 July 2021. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[82] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE. 2009, pp. 248–255.

[83] Claire Adam-Bourdarios et al. “The Higgs boson machine learning challenge”. In: *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*. Ed. by Glen Cowan et al. Vol. 42. Proceedings of Machine Learning Research. Montreal, Canada: PMLR, 13 Dec 2015, pp. 19–55. URL: http://proceedings.mlr.press/v42/cowa14.html.

[84] Sabrina Amrouche et al. “The Tracking Machine Learning challenge : Accuracy phase”. In: (Apr. 2019). DOI: 10.1007/978-3-030-29135-8_9. arXiv: 1904.06778 [hep-ex].

[85] Anja Butter et al. “The Machine Learning Landscape of Top Taggers”. In: *SciPost Phys.* 7 (2019). Ed. by Gregor Kasieczka and Tilman Plehn, p. 014. DOI: 10.21468/SciPostPhys.7.1.014. arXiv: 1902.09914 [hep-ph].

[86] David Rousseau and Andrey Ustyuzhanin. “Machine Learning scientific competitions and datasets”. In: (Dec. 2020). arXiv: 2012.08520 [physics.data-an].

[87] Leandro G. Almeida et al. “Playing Tag with ANN: Boosted Top Identification with Pattern Recognition”. In: *JHEP* 07 (2015), p. 086. DOI: 10.1007/JHEP07(2015)086. arXiv: 1501.05968 [hep-ph].

[88] Shannon Egan et al. “Long Short-Term Memory (LSTM) networks with jet constituents for boosted top tagging at the LHC”. In: (Nov. 2017). arXiv: 1711.09059 [hep-ex].

[89] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. "Energy Flow Networks: Deep Sets for Particle Jets". In: *JHEP* 01 (2019), p. 121. DOI: 10.1007/JHEP01(2019)121. arXiv: 1810.05165 [hep-ph].

[90] Huilin Qu and Loukas Gouskos. "ParticleNet: Jet Tagging via Particle Clouds". In: *Phys. Rev. D* 101.5 (2020), p. 056019. DOI: 10.1103/PhysRevD.101.056019. arXiv: 1902.08570 [hep-ph].

[91] Emil Bols et al. "Jet Flavour Classification Using DeepJet". In: (Aug. 2020). DOI: 10.1088/1748-0221/15/12/P12012. arXiv: 2008.10519 [hep-ex].

[92] Lisa Benato et al. "Shared Data and Algorithms for Deep Learning in Fundamental Physics". In: *Comput. Softw. Big Sci.* 6.1 (2022), p. 9. DOI: 10.1007/s41781-022-00082-6. arXiv: 2107.00656 [cs.LG].

[93] Kahn, James et al. "Selective background Monte Carlo simulation at Belle II". In: *EPJ Web Conf.* 245 (2020), p. 02028. DOI: 10.1051/epjconf/202024502028. URL: https://doi.org/10.1051/epjconf/202024502028.

[94] Jan Steinheimer et al. "A machine learning study to identify spinodal clumping in high energy nuclear collisions". In: *JHEP* 12 (2019), p. 122. DOI: 10.1007/JHEP12(2019)122. arXiv: 1906.06562 [nucl-th].

[95] Long-Gang Pang et al. "Classify QCD phase transition with deep learning". In: *Nucl. Phys.* A982 (2019), pp. 867–870. DOI: 10.1016/j.nuclphysa.2018.10.077.

[96] Gregor Kasieczka et al. *Top Quark Tagging Reference Dataset*. Version v0 (2018_03_27). Zenodo, Mar. 2019. DOI: 10.5281/zenodo.2603256. URL: https://doi.org/10.5281/zenodo.2603256.

[97] Torbjörn Sjöstrand et al. "An introduction to PYTHIA 8.2". In: *Comput. Phys. Commun.* 191 (2015), pp. 159–177. DOI: 10.1016/j.cpc.2015.01.024. arXiv: 1410.3012 [hep-ph].

[98] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "The anti-k_t jet clustering algorithm". In: *JHEP* 04 (2008), p. 063. DOI: 10.1088/1126-6708/2008/04/063. arXiv: 0802.1189 [hep-ph].

[99] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "FastJet User Manual". In: *Eur. Phys. J. C* 72 (2012), p. 1896. DOI: 10.1140/epjc/s10052-012-1896-2. arXiv: 1111.6097 [hep-ph].

[100] James Meier Samuel Kahn. "Hadronic tag sensitivity study of B $\to$ K(*)$\nu\bar{\nu}$ and selective background Monte Carlo Simulation at Belle II". PhD thesis. Apr. 2019. URL: http://nbn-resolving.de/urn:nbn:de:bvb:19-240131.

[101] D. J. Lange. "The EvtGen particle decay simulation package". In: *Nucl. Instrum. Meth. A* 462 (2001). Ed. by S. Erhan, P. Schlein, and Y. Rozen, pp. 152–155. DOI: 10.1016/S0168-9002(01)00089-4.

[102] Particle Data Group. "Review of Particle Physics". In: *PTEP* 2020.8 (2020), p. 083C01. DOI: 10.1093/ptep/ptaa104.

[103] T. Keck et al. "The Full Event Interpretation: An Exclusive Tagging Algorithm for the Belle II Experiment". In: *Comput. Softw. Big Sci.* 3.1 (2019), p. 6. DOI: 10.1007/s41781-019-0021-8. arXiv: 1807.08680 [hep-ex].

[104] Jan Steinheimer. *Spinodal Dataset for classification*. Zenodo, Nov. 2021. DOI: 10.5281/zenodo.5710737. URL: https://doi.org/10.5281/zenodo.5710737.

[105] Jan Steinheimer et al. "A machine learning study on spinodal clumping in heavy ion collisions". In: *Nucl. Phys. A* 1005 (2021). Ed. by Feng Liu et al., p. 121867. DOI: 10.1016/j.nuclphysa.2020.121867.

[106] Long-Gang Pang et al. "An equation-of-state-meter of quantum chromodynamics transition from deep learning". In: *Nature Commun.* 9.1 (2018), p. 210. DOI: 10.1038/s41467-017-02726-3. arXiv: 1612.04262 [hep-ph].

[107] Yi-Lun Du et al. "Identifying the nature of the QCD transition in relativistic collision of heavy nuclei with deep learning". In: *Eur. Phys. J.* C80.6 (2020), p. 516. DOI: 10.1140/epjc/s10052-020-8030-7. arXiv: 1910.11530 [hep-ph].

[108] Yi-Lun Du et al. "Identifying the nature of the QCD transition in heavy-ion collisions with deep learning". In: *Nucl. Phys.* A1005 (2021), p. 121891. DOI: 10.1016/j.nuclphysa.2020.121891. arXiv: 2009.03059 [nucl-th].

[109] Jonas Glombitza. *Reconstruction of simulated air shower footprints measured at a hypothetical ground-based cosmic-ray observatory*. Zenodo, Dec. 2021. DOI: 10.5281/zenodo.5748080. URL: https://doi.org/10.5281/zenodo.5748080.

[110] M. Erdmann, J. Glombitza, and D. Walz. "A deep learning-based reconstruction of cosmic ray-induced air showers". In: *Astroparticle Physics* 97 (2018), pp. 46–53. ISSN: 0927-6505. DOI: https://doi.org/10.1016/j.astropartphys.2017.10.006. URL: https://www.sciencedirect.com/science/article/pii/S0927650517302219.

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

[112] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[113] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2015). arXiv: 1412.6980 [cs.LG].

[114] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: (2017). arXiv: 1609.02907 [cs.LG].

[115] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (June 2014), pp. 1929–1958.

[116] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *CoRR* abs/1502.01852 (2015). arXiv: 1502.01852. URL: http://arxiv.org/abs/1502.01852.

[117] Huilin Qu and Loukas Gouskos. "Jet tagging via particle clouds". In: *Physical Review D* 101.5 (Mar. 2020). ISSN: 2470-0029. DOI: 10.1103/physrevd.101.056019. URL: http://dx.doi.org/10.1103/PhysRevD.101.056019.

[118] Yue Wang et al. “Dynamic Graph CNN for Learning on Point Clouds”. In: (2018). arXiv: 1801.07829 [cs.CV].

[119] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (2015). arXiv: 1409.1556 [cs.CV].

[120] The Pierre Auger Collaboration. “Deep-Learning based Reconstruction of the Shower Maximum $X_{\max}$ using the Water-Cherenkov Detectors of the Pierre Auger Observatory”. In: (Jan. 8, 2021). arXiv: 2101.02946 [astro-ph, physics:hep-ex].

[121] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[122] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: (Dec. 10, 2015). arXiv: 1512.03385 [cs].

[123] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: (Mar. 2, 2015). arXiv: 1502.03167 [cs].

[124] CMS. *CMS Physics TDR*. 2 February 2006.

[125] S. Agostinelli et al. “GEANT4–a simulation toolkit”. In: *Nucl. Instrum. Meth. A* 506 (2003), p. 250. DOI: 10.1016/S0168-9002(03)01368-8.

[126] Erik Buhmann et al. “Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed”. In: *Comput. Softw. Big Sci.* 5.1 (2021), p. 13. DOI: 10.1007/s41781-021-00056-0. eprint: 2005.05334.

[127] Erik Buhmann et al. “Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network”. In: *EPJ Web Conf.* 251 (2021), p. 03003. DOI: 10.1051/epjconf/202125103003. eprint: 2102.12491.

[128] Erik Buhmann et al. “Hadrons, better, faster, stronger”. In: *Mach. Learn. Sci. Tech.* 3.2 (2022), p. 025014. DOI: 10.1088/2632-2153/ac7848. arXiv: 2112.09709 [physics.ins-det].

[129] ATLAS Collaboration. *Deep generative models for fast photon shower simulation in ATLAS*. 2022. arXiv: 2210.06204 [hep-ex].

[130] Jesse C. Cresswell et al. *CaloMan: Fast generation of calorimeter showers with density estimation on learned manifolds*. 2022. arXiv: 2211.15380 [hep-ph].

[131] Sascha Diefenbacher et al. “New Angles on Fast Calorimeter Shower Simulation”. In: (Mar. 2023). arXiv: 2303.18150 [physics.ins-det].

[132] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. “Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters”. In: *Phys. Rev. Lett.* 120.4 (2018), p. 042003. DOI: 10.1103/PhysRevLett.120.042003. arXiv: 1705.02355 [hep-ex].

[133] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. “CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks”. In: *Phys. Rev. D* 97.1 (2018), p. 014021. DOI: 10.1103/PhysRevD.97.014021. arXiv: 1712.10321 [hep-ex].

[134] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. "Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters". In: *J. Phys. Conf. Ser.* 1085.4 (2018), p. 042017. DOI: 10.1088/1742-6596/1085/4/042017. arXiv: 1711.08813 [hep-ex].

[135] Martin Erdmann et al. "Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks". In: *Comput. Softw. Big Sci.* 2.1 (2018), p. 4. DOI: 10.1007/s41781-018-0008-x. arXiv: 1802.03325 [astro-ph.IM].

[136] Martin Erdmann, Jonas Glombitza, and Thorben Quast. "Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network". In: *Comput. Softw. Big Sci.* 3.1 (2019), p. 4. DOI: 10.1007/s41781-018-0019-7. arXiv: 1807.01954 [physics.ins-det].

[137] Dawit Belayneh et al. "Calorimetry with deep learning: particle simulation and reconstruction for collider physics". In: *The European Physical Journal C* 80.7 (July 2020). DOI: 10.1140/epjc/s10052-020-8251-9. URL: https://doi.org/10.1140%2Fepjc%2Fs10052-020-8251-9.

[138] *Fast simulation of the ATLAS calorimeter system with Generative Adversarial Networks*. Tech. rep. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-SOFT-PUB-2020-006. Geneva: CERN, 2020. URL: https://cds.cern.ch/record/2746032.

[139] F. Carminati et al. "Three dimensional Generative Adversarial Networks for fast simulation". In: *J. Phys. Conf. Ser.* 1085.3 (2018), p. 032016. DOI: 10.1088/1742-6596/1085/3/032016.

[140] Pasquale Musella and Francesco Pandolfi. "Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks". In: *Comput. Softw. Big Sci.* 2.1 (2018), p. 8. DOI: 10.1007/s41781-018-0015-y. arXiv: 1805.00850 [hep-ex].

[141] The ATLAS collaboration. *Deep generative models for fast shower simulation in ATLAS*. Tech. rep. Geneva: CERN, 2018. URL: http://cds.cern.ch/record/2630433.

[142] ATLAS Collaboration. "AtlFast3: The Next Generation of Fast Simulation in ATLAS". In: *Comput. Softw. Big Sci.* 6.7 (2022). DOI: https://doi.org/10.1007/s41781-021-00079-7.

[143] Hosein Hashemi et al. "Ultra-High-Resolution Detector Simulation with Intra-Event Aware GAN and Self-Supervised Relational Reasoning". In: (Mar. 2023). arXiv: 2303.08046 [physics.ins-det].

[144] C. Chen et al. "Analysis-Specific Fast Simulation at the LHC with Deep Learning". In: *Computing and Software for Big Science* 5.1 (2021), p. 15. DOI: 10.1007/s41781-021-00060-4. URL: https://doi.org/10.1007/s41781-021-00060-4.

[145] Claudius Krause and David Shih. "CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows". In: (June 2021). DOI: 10.48550/arXiv.2106.05285. arXiv: 2106.05285 [physics.ins-det].

[146] Claudius Krause and David Shih. "CaloFlow II: Even Faster and Still Accurate Generation of Calorimeter Showers with Normalizing Flows". In: (2021). DOI: 10.48550/arXiv.2110.11377. arXiv: 2110.11377 [physics.ins-det].

[147] Simon Schnake, Dirk Krücker, and Kerstin Borras. *Generating Calorimeter Showers as Point Clouds*. Dec. 2022. URL: https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_77.pdf.

[148] Claudius Krause, Ian Pang, and David Shih. *CaloFlow for CaloChallenge Dataset 1*. 2023. arXiv: 2210.14245 [physics.ins-det].

[149] Sascha Diefenbacher et al. "L2LFlows: Generating High-Fidelity 3D Calorimeter Images". In: (2023). arXiv: 2302.11594 [physics.ins-det].

[150] Allison Xu et al. *Generative Machine Learning for Detector Response Modeling with a Conditional Normalizing Flow*. 2023. arXiv: 2303.10148 [hep-ex].

[151] Matthew R. Buckley et al. *Inductive CaloFlow*. 2023. arXiv: 2305.11934 [physics.ins-det].

[152] Erik Buhmann et al. "EPiC-ly Fast Particle Cloud Generation with Flow-Matching and Diffusion". In: (Sept. 2023). arXiv: 2310.00049 [hep-ph].

[153] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG].

[154] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2020. arXiv: 1907.05600 [cs.LG].

[155] Yang Song and Stefano Ermon. *Improved Techniques for Training Score-Based Generative Models*. 2020. arXiv: 2006.09011 [cs.LG].

[156] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[157] Yang Song et al. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: 2011.13456 [cs.LG].

[158] Vinicius Mikuni and Benjamin Nachman. "Score-based generative models for calorimeter shower simulation". In: *Phys. Rev. D* 106.9 (2022), p. 092009. DOI: 10.1103/PhysRevD.106.092009. arXiv: 2206.11898 [hep-ph].

[159] Erik Buhmann et al. *CaloClouds: Fast Geometry-Independent Highly-Granular Calorimeter Simulation*. 2023. arXiv: 2305.04847 [physics.ins-det].

[160] Erik Buhmann et al. "CaloClouds II: ultra-fast geometry-independent highly-granular calorimeter simulation". In: *JINST* 19.04 (2024), P04020. DOI: 10.1088/1748-0221/19/04/P04020. arXiv: 2309.05704 [physics.ins-det].

[161] Fernando Torales Acosta et al. *Comparison of Point Cloud and Image-based Models for Calorimeter Fast Simulation*. 2023. arXiv: 2307.04780 [cs.LG].

[162] Vinicius Mikuni and Benjamin Nachman. *CaloScore v2: Single-shot Calorimeter Shower Simulation with Diffusion Models*. 2023. arXiv: 2308.03847 [hep-ph].

[163] Oz Amram and Kevin Pedro. *CaloDiffusion with GLaM for High Fidelity Calorimeter Simulation*. 2023. arXiv: 2308.03876 [physics.ins-det].

[164] Halina Abramowicz et al. "The International Linear Collider Technical Design Report - Volume 4: Detectors". In: (June 2013). Ed. by Ties Behnke et al. arXiv: 1306.6329 [physics.ins-det].

[165] Ties Behnke et al. “The International Linear Collider Technical Design Report - Volume 1: Executive Summary”. In: (June 2013). DOI: 10.48550/arXiv.1306.6327. arXiv: 1306.6327 [physics.acc-ph].

[166] J. S. Marshall and M. A. Thomson. “The Pandora software development kit for pattern recognition”. In: *The European Physical Journal C* 75.9 (Sept. 2015). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-015-3659-3. eprint: 1506.05348.

[167] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[168] Erik Buhmann, Gregor Kasieczka, and Jesse Thaler. “EPiC-GAN: Equivariant point cloud generation for particle jets”. In: *SciPost Physics* 15.4 (Oct. 2023). ISSN: 2542-4653. DOI: 10.21468/scipostphys.15.4.130. URL: http://dx.doi.org/10.21468/SciPostPhys.15.4.130.

[169] James M. Joyce. “Kullback-Leibler Divergence”. In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327. URL: https://doi.org/10.1007/978-3-642-04898-2_327.

[170] Anja Butter, Tilman Plehn, and Ramon Winterhalder. “How to GAN LHC Events”. In: *SciPost Phys.* 7.6 (2019), p. 075. DOI: 10.21468/SciPostPhys.7.6.075. arXiv: 1907.03764 [hep-ph].

[171] Bobak Hashemi et al. “LHC analysis-specific datasets with Generative Adversarial Networks”. In: (Jan. 2019). arXiv: 1901.05282 [hep-ex].

[172] Riccardo Di Sipio et al. “DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC”. In: *JHEP* 08 (2019), p. 110. DOI: 10.1007/JHEP08(2019)110. arXiv: 1903.02433 [hep-ex].

[173] Jesús Arjona Martínez et al. “Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description”. In: *J. Phys. Conf. Ser.* 1525.1 (2020), p. 012081. DOI: 10.1088/1742-6596/1525/1/012081. arXiv: 1912.02748 [hep-ex].

[174] Yasir Alanazi et al. “Simulation of Electron-Proton Scattering Events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. IJCAI, Aug. 2021, p. 2126. DOI: 10.24963/ijcai.2021/293. eprint: 2001.11103.

[175] Sydney Otten et al. “Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer”. In: *Nature Commun.* 12.1 (2021), p. 2985. DOI: 10.1038/s41467-021-22616-z. arXiv: 1901.00875 [hep-ph].

[176] Anja Butter et al. “Generative Networks for Precision Enthusiasts”. In: (Oct. 2021). arXiv: 2110.13632 [hep-ph].

[177] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. "Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis". In: *Computing and Software for Big Science* 1.1 (Sept. 2017). DOI: 10.1007/s41781-017-0004-6. URL: https://doi.org/10.1007%2Fs41781-017-0004-6.

[178] Anders Andreassen et al. "JUNIPR: a framework for unsupervised machine learning in particle physics". In: *The European Physical Journal C* 79.2 (Feb. 2019). DOI: 10.1140/epjc/s10052-019-6607-9. URL: https://doi.org/10.1140%2Fepjc%2Fs10052-019-6607-9.

[179] Enrico Bothmann and Luigi Del Debbio. "Reweighting a parton shower using a neural network: the final-state case". In: *Journal of High Energy Physics* 2019.1 (Jan. 2019). DOI: 10.1007/jhep01(2019)033. URL: https://doi.org/10.1007%2Fjhep01%282019%29033.

[180] Kosei Dohi. "Variational Autoencoders for Jet Simulation". In: (Sept. 2020). arXiv: 2009.04842 [hep-ph].

[181] Raghav Kansal et al. *Particle Cloud Generation with Message Passing Generative Adversarial Networks*. 2022. arXiv: 2106.11535 [cs.LG].

[182] Benno Käch et al. "JetFlow: Generating Jets with Conditioned and Mass Constrained Normalising Flows". In: (2022). arXiv: 2211.13630 [hep-ex].

[183] Benno Käch, Dirk Krücker, and Isabell Melzer-Pellmann. *Point Cloud Generation using Transformer Encoders and Normalising Flows*. 2022. arXiv: 2211.13623 [hep-ex].

[184] Raghav Kansal et al. "Evaluating generative models in high energy physics". In: *Physical Review D* 107.7 (Apr. 2023). DOI: 10.1103/physrevd.107.076017. URL: https://doi.org/10.11032Fphysrevd.107.076017.

[185] Erik Buhmann, Gregor Kasieczka, and Jesse Thaler. "EPiC-GAN: Equivariant Point Cloud Generation for Particle Jets". In: (Jan. 2023). arXiv: 2301.08128 [hep-ph].

[186] Matthew Leigh et al. *PC-JeDi: Diffusion for Particle Cloud Generation in High Energy Physics*. 2023. arXiv: 2303.05376 [hep-ph].

[187] Benno Käch and Isabell Melzer-Pellmann. *Attention to Mean-Fields for Particle Cloud Generation*. 2023. arXiv: 2305.15254 [hep-ex].

[188] Anja Butter et al. *Jet Diffusion versus JetGPT – Modern Networks for the LHC*. 2023. arXiv: 2305.10475 [hep-ph].

[189] Matthew Leigh et al. *PC-Droid: Faster diffusion and improved quality for particle cloud generation*. 2023. arXiv: 2307.06836 [hep-ex].

[190] Baran Hashemi and Claudius Krause. "Deep generative models for detector signature simulation: A taxonomic review". In: *Rev. Phys.* 12 (2024), p. 100092. DOI: 10.1016/j.revip.2024.100092. arXiv: 2312.09597 [physics.ins-det].

[191] Erik Buhmann et al. "EPiC-ly Fast Particle Cloud Generation with Flow-Matching and Diffusion". In: (Sept. 2023). arXiv: 2310.00049 [hep-ph].

[192] My Boy et al. “Flow Matching Beyond Kinematics: Generating Jets with Particle-ID and Trajectory Displacement Information”. In: (Nov. 2023). arXiv: 2312.00123 [hep-ph].

[193] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML].

[194] Shitong Luo and Wei Hu. *Diffusion Probabilistic Models for 3D Point Cloud Generation*. 2021. arXiv: 2103.01458 [cs.CV].

[195] Tero Karras et al. *Elucidating the Design Space of Diffusion-Based Generative Models*. 2022. arXiv: 2206.00364 [cs.CV].

[196] Will Grathwohl et al. *FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models*. 2018. arXiv: 1810.01367 [cs.LG].

[197] Rene Brun et al. *root-project/root: v6.18/02*. Version v6-18-02. Aug. 2019. DOI: 10.5281/zenodo.3895860.

[198] Ranit Das et al. *How to Understand Limitations of Generative Networks*. 2023. arXiv: 2305.16774 [hep-ph].

[199] Mark Rygielski. “Improving generative fidelity on high granularity data using the attention mechanism”. University of Hamburg, 2025.