Generative Machine Learning for Fast Particle Physics Simulations

Dissertation

zur Erlangung des Doktorgrades

an der Fakultät für Mathematik, Informatik und Naturwissenschaften

Fachbereich Physik

der Universität Hamburg

vorgelegt von

Erik Buhmann

Hamburg

2024

Gutachter der Dissertation:	Prof. Dr. Gregor Kasieczka Dr. Frank Gaede
Zusammensetzung der Prüfungskommission:	Prof. Dr. Gregor Kasieczka Dr. Frank Gaede Prof. Dr. Erika Garutti Prof. Dr. Marcus Brüggen Prof. Dr. Daniela Pfannkuche
Vorsitzende der Prüfungskommission:	Prof. Dr. Daniela Pfannkuche
Datum der Disputation:	29.01.2025
Vorsitzender 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

Zusammenfassung

Die Suche nach Physik jenseits des Standardmodells ist ein zentrales Ziel der Teilchenphysik. Diese Forschung wird an Teilchenbeschleuniger-Experimenten durchgeführt und erfordert eine große Menge an simulierten Daten. Der Ausbau des Large Hadron Colliders (LHC) zum High-Luminosity LHC steigert den Bedarf an schnellen Simulationen erheblich. Für den HL-LHC wird der CMS-Detektor mit hochgranularen Endkappenkalorimetern ausgestattet, und ähnliche hochgranulare Kalorimeter sind auch für zukünftige Teilchendetektoren vorgesehen. Zusammengenommen erhöhen diese Faktoren den Bedarf an präziseren schnellen Simulationen. In dieser Arbeit wird generatives maschinelles Lernen als Werkzeug für sehr präzise schnelle Simulationen untersucht.

Es werden mehrere Modelle für die schnelle Simulation von Kalorimeterschauern und Jets vorgestellt. Das Bounded Information Bottleneck Autoencoder (BIB-AE) Modell erzeugt Kalorimeterschauer als 3-dimensionale Bilder. Eine Analyse des kodierten latenten Raums des BIB-AE zeigt, dass nur wenige Variablen die meisten Schauerinformationen kodieren. Dies motiviert eine Verbesserung des BIB-AE durch den Einsatz eines Kernel Density Estimators zur Modellierung des latenten Raums. Das resultierende Modell ist in der Lage, hochgranulare Photonenschauer mit hoher Genauigkeit 10-mal schneller zu simulieren als die herkömmliche Monte-Carlo-Simulation Geant4 auf der gleichen CPU Hardware.

Um die Effizienz der generativen Modelle für Kalorimeterschauer weiter zu steigern, wird das Diffusionsmodell CaloClouds eingeführt, welches Kalorimeterschauer als Punktwolken modelliert. Diese Darstellung bietet mehrere Vorteile gegenüber 3D-Bildern; unter anderem ist sie effizienter und ermöglicht eine geometrieunabhängige Schauermodellierung. Das CaloClouds II Model verbessert diesen Ansatz mittels kontinuierlichem Score-Matching und erreicht eine höhere Genauigkeit und schnellere Generierung. Das Modell wird weiter zum Consistency Model CaloClouds II (CM) destilliert, was nicht nur die Geschwindigkeit erheblich erhöht, sondern auch die Genauigkeit weiter verbessert. CaloClouds II (CM) ist auf der gleichen Hardware 46-mal schneller als Geant4.

Als letztes wird der Equivariant Point Cloud (EPiC) Netzerk-Layer eingeführt, um die in der Teilchenphysik verwendeten generativen Punktwolkenmodelle weiter zu verbessern. Der Layer wird in drei verschiedenen generativen Punktwolkenmodellen verwendet: im adversiellen generativen Netzwerk EPiC-GAN, im Diffusionsmodell EPiC-JeDi und im Continuous Normalizing Flow Modell EPiC-FM, welches mittels Flow-Matching trainiert ist. Die Modelle werden anhand des JetNet Datensatzes für die Erzeugung von Teilchenjets bewertet. Dabei zeigt sich, dass EPiC-GAN das effizienteste Modell ist, da es etwa 210-mal schneller ist als die anderen beiden Modelle und immer noch die Leistung des komplexeren, graphbasierten MP-GAN erreicht. EPiC-FM ist jedoch das genaueste aller verglichenen Modelle, was den Flow-Matching-Ansatz und die EPiC Layer als vielversprechende Ansätze für zukünftige generative Modelle für schnelle Simulationen in der Teilchenphysik hervorhebt.

Abstract

The search for physics beyond the Standard Model is a central goal of particle physics. This research is conducted at collider experiments and requires a very large amount of simulated data. With the high-luminosity upgrade to the Large Hadron Collider (HL-LHC) the need for more and faster simulations is increasing. The CMS detector upgrade for the HL-LHC will feature high-granularity endcap calorimeters and highly granular calorimeters are also envisioned to be used at future collider detectors. Together, these factors heighten the demand for more precise fast simulations. In this thesis, generative machine learning is explored as a tool for high fidelity fast simulations.

Several models for the fast simulation of calorimeter showers and jets are presented. The bounded information bottleneck autoencoder (BIB-AE) model generates calorimeter showers as 3-dimensional images. Its encoded latent space is analyzed and it is shown that only few variables encode most shower information. This motivates an improvement of the BIB-AE using a kernel density estimator to model the latent space. The resulting model is able to simulate highly granular photon showers with high fidelity at 10x faster than the traditional Monte Carlo simulation Geant4 on the same CPU hardware.

To advance calorimeter shower generative models in terms of computational efficiency, the diffusion model CaloClouds, which models calorimeter shower as point clouds, is introduced. The representation as point clouds has several advantages over 3D images, including being more efficient and allowing for a geometry-independent shower modeling. The CaloClouds II model improves the approach by applying continuous-time score matching to achieve a higher fidelity and faster generation. The model is further distilled into the consistency model CaloClouds II (CM) which not only greatly accelerates the model, it also increases the fidelity further. CaloClouds II (CM) is 46x faster than Geant4 on the same hardware.

Finally, the equivariant point cloud (EPiC) layer structure is introduced to further improve point cloud generative models used in particle physics. The layer is utilized in three different point cloud generative models: in the generative adversarial network EPiC-GAN, in the score-based diffusion model EPiC-JeDi, and in the continuous normalizing flow EPiC-FM, trained with the flow matching objective. The models are evaluated on the common JetNet benchmark dataset for the generation of particle jets. The EPiC-GAN is the most efficient model being about 210x faster than the other two models and still reaches the performance of the more complex previous state-of-the-art graph-based MP-GAN. However, EPiC-FM is the most accurate among all compared models. This underscores the flow matching approach and the EPiC layer structure as promising directions for future generative model for fast simulations in particle physics.

Contents

Preface

The findings presented in this thesis are the result of research conducted by the author in collaboration with other researchers, in the time between 2019 and 2024 at the Institute for Experimental Physics at the University of Hamburg. Most of the results shown in Chapters 5, 6, and 7 have previously been published as:

[1]: E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, and K. Krüger 2021 **Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network** *EPJ Web Conf.* **251** 03003. 10.1051/epjconf/202125103003

[2]: E. Buhmann, G. Kasieczka, and J. Thaler 2023 **EPiC-GAN: Equivariant point cloud generation for particle jets** *SciPost Phys.* **15** 130. 0.21468/SciPostPhys.15.4.130

[3]: E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger, and P. McKeown 2023 **CaloClouds: fast geometry-independent highly-granular calorimeter simulation** *JINST* **18** P11025. 10.1088/1748-0221/18/11/P11025

[4]: E. Buhmann, C. Ewen, D. A. Faroughy, T. Golling, G. Kasieczka, M. Leigh, G. Quétant, J. A. Raine, D. Sengupta, and D. Shih 2024 **EPiC-ly Fast Particle Cloud Generation with Flow-Matching and Diffusion** *Submitted to EPJ C.* 10.48550/arXiv.2310.00049

[5]: E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger, and P. McKeown 2024 **CaloClouds II: ultra-fast geometry-independent highly-granular calorimeter simulation** *JINST* **19** P04020. 10.1088/1748-0221/19/04/P04020

Additionally, the author has contributed to the following publications during this research period:

[6]: E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczk, A. Korol, and K. Krüger 2021 **Getting high: High fidelity simulation of high granularity calorimeters with high speed** *Comput. Softw. Big Sci.* **5** 13. 10.1007/s41781-021-00056-0

[7]: E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, W. Korcari, A. Korol, K. Krüger, P. McKeown, and L. Rustige 2021 **Fast and Accurate Electromagnetic and Hadronic Showers from Generative Models** *EPJ Web Conf.* textbf251 03049. 10.1051/epjconf/202125103049

[8]: L. Benato, E. Buhmann, M. Erdmann, P. Fackeldey, J. Glombitza, N. Hartmann, G. Kasieczka, W. Korcari, T. Kuhr, J. Steinheimer, H. Stöcker, T. Plehn, and K. Zhou 2022

Shared Data and Algorithms for Deep Learning in Fundamental Physics *Comput Softw Big Sci* **6** 9. 10.1007/s41781-022-00082-6

[9]: E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, W. Korcari, K. Krüger, P. McKeown, and L. Rustige 2022 **Hadrons, better, faster, stronger** *Mach. Learn. Sci. Tech.* **3** 025014. 10.1088/2632-2153/ac7848

[10]: P. McKeown, S. Bieringer, E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, W. Korcari, A. Korol, K. Krüger, L. Rustige and I. Shekhzadeh 2022 **Generative Models for Fast Simulation of Electromagnetic and Hadronic Showers in Highly Granular Calorimeters** *PoS ICHEP2022.* 10.22323/1.414.0236

[11]: J. Birk, E. Buhmann, C. Ewen, G. Kasieczka, and D. Shih 2023 **Flow Matching Beyond Kinematics: Generating Jets with Particle-ID and Trajectory Displacement Information** *Submitted to Phys. Rev. D.* 10.48550/arXiv.2312.00123

[12]: E. Buhmann, C. Ewen, G. Kasieczka, V. Mikuni, B. Nachman, and D. Shih 2024 **Full phase space resonant anomaly detection** *Phys. Rev. D* **109** 055015. 10.1103/PhysRevD.109.055015

[13]: P. McKeown, E. Buhmann, T. Buss, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, W. Korcari, A. Korol, K. Krüger, T. Madlener, and L. Rustige 2023 **Fast Simulation of Highly Granular Calorimeters with Generative Models: Towards a First Physics Application** *PoS EPS-HEP2023.* 10.22323/1.449.0568

and to the following publication in preparation:

[14]: C. Krause, M. F. Giannelli, G Kasieczka, *et al.* 2024 **CaloChallenge 2022: A Community Challenge for Fast Calorimeter Simulation** *In preparation.*

Finally, the author supervised the following thesis projects during this time:

[15]: M. Jacobsen 2021 **The Blessing of Dimensionality: Event Manifold Dimensionality Estimation for Event Clustering** *Bachelor thesis at the University of Hamburg.*

[16]: N. M. Werther 2022 **Frechet Distance Evaluation of Generative Models for Calorimeter Shower Simulation** *Bachelor thesis at the University of Hamburg.*

[17]: J. Schreiber 2023 **Evaluation of Evaluation Metrics of Generative Models in High Energy Physics** *Bachelor thesis at the University of Hamburg.*

[18]: C. Ewen 2023 **Frechet Distance Evaluation of Generative Models for Calorimeter Shower Simulation** *Master thesis at the University of Hamburg.*

Introduction

The fundamental goal of physics is to understand the laws of nature that govern the universe. In particle physics, the elementary building blocks of matter and the forces that act between them are studied. The Standard Model of particle physics is currently the most comprehensive description of our understanding of nature at the smallest scales. It describes all known elementary particles and the mediators of the electromagnetic, weak, and strong forces. The latest success of the Standard Model is the discovery of the Higgs boson [19,20] by the ATLAS and CMS experiments at the Large Hadron Collider (LHC) at CERN.

While the Standard Model is among the most successful theories in physics, it has multiple shortcomings which motivate the search for physics beyond the Standard Model. For example, it does not include a description of gravity, and it does not explain observed astronomical phenomena that are attributed to dark matter and dark energy. Experiments at particle colliders such as the LHC are crucial to probe the SM and search for new physics. Soon, the LHC will be upgraded to the High-Luminosity LHC (HL-LHC) [21], which will provide a five times increase in instantaneous luminosity and thereby enable further precision studies at the energy frontier. Several detector components will be upgraded to maintain a high level of precision in an environment with much higher collision rates. This includes for example the CMS endcap calorimeters, which will be replaced with high-granularity calorimeters [22].

To continue precision measurements and searches for new physics, after the HL-LHC physics program is concluded, the particle physics community is planning the construction of a future lepton collider as a Higgs factory [23]. A lepton collider would allow for much more precise measurements of the Higgs boson and other particles than are currently possible at the LHC. To separate well the hadronic decays of the $W^{\pm}$ and Z bosons into multi-jet final states at such a collider, detector experiments are aiming for a jet energy resolution of 3-4% in the range of about 45–250 GeV [24]. This can be achieved by applying Particle Flow Algorithms that optimize which detector sub-system is used to reconstruct a given particle [25]. To separate particles for the application of the PF algorithms, an excellent tracking system and highly-granular calorimeters are required.

In particle physics, simulations provide an important bridge between theory and experiment. For the analyses at detector experiments, at least as many simulated events as real data events are needed, so when the HL-LHC luminosity rises, so does the required amount of simulated events. Full Monte Carlo simulations are computationally expensive, so fast simulations are used where possible. These fast simulations trade some precision for speed, but with the advent of highly-granular calorimeters, the required fidelity of fast simulations is increasing. Both factors together — the increasing collider luminosity and the increasing

granularity of the detectors — warrant the development of more precise fast simulations.

The Standard Model of particle physics and the physics beyond the Standard Model that motivates the search for new physics are introduced in Chapter 1. It discusses the experiments at current and future collider experiments, explains the importance of accurate simulations, and introduces the study of jet physics. In Chapter 2, the principles of particle-matter interaction and the formation of particle showers in calorimeters are discussed. Furthermore, an introduction to the particle flow approach for jet reconstruction and to traditional and fast calorimeter simulations is given.

In recent years, interest has grown in the application of generative machine learning models for precise and fast simulations. Generative models employ deep neural networks with potentially millions of parameters to learn a probability distribution from a given dataset. A well-trained generative model can then generate new samples from this distribution that are very similar to the training data. In high-energy physics (HEP), generative models[1] have been investigated as fast simulations tools for various tasks, including event generation [27–38] and detector simulation [1, 3, 5, 6, 9, 39–61]. The latter is of particular importance, since the detector simulation is the most time-consuming part of the simulation chain.

The basics of machine learning are explained in Chapter 3. It is discussed how machine learning models are optimized and how they can be parameterized as deep neural networks. Two neural network concepts are specifically discussed: Convolutional neural networks, that are optimal for image data, and graph- and set-based neural networks, that are optimal for data without a fixed structure such as many measurements at detector experiments. Chapter 4 introduces five different types of generative models that are utilized in this thesis for fast simulation studies. These include generative adversarial networks, autoencoders, normalizing flows, diffusion models, and consistency models.

Most generative models for calorimeter simulation model the shower as a 3D image to be able to use established convolutional neural network architectures. In Chapter 5 an analysis of such a model is presented. Specifically, the encoded latent space in the bounded information bottleneck autoencoder (BIB-AE) model is studied to understand what kind of physics the model learns. The BIB-AE models high-granularity photon showers in the electromagnetic calorimeter of the envisioned International Large Detector (ILD) at the International Linear Collider (ILC) [24]. The information-theoretical analysis motivates recommendations for improvements of the BIB-AE model. Additionally, several methods for the evaluation of generative models are evaluated.

Although a high fidelity is reached with the voxelized modeling of calorimeter showers, the approach is not the most efficient for the simulation of highly-granular calorimeters. In a typical shower, energy is deposited only in a few percent of the calorimeter cells, i.e. only a few voxels of the 3D image are non-zero. Hence, generative model produces mostly empty images, which is computationally inefficient. A better approach would be to model calorimeter showers as point clouds, where each point represents a cell with an energy deposition. To this end, the CaloClouds model family is introduced in Chapter 6. These models are designed to simulate calorimeter showers as high-cardinality point clouds. Three models are presented that constitute an evolution in the model design. They are evaluated in their generative

[1]The references listed in the following are not exhaustive. A living review can be found in Reference [26].

performance and their simulation speed on high-granularity photon showers.

While the approach of modeling calorimeter showers as point clouds is more efficient than the voxelized approach, it requires a special type of neural network architecture that can handle data without a fixed structure. In Chapter 7, a novel neural network layer specifically tailored to the modeling of point clouds is presented. This layer, dubbed equivariant point cloud (EPiC) layer, is designed to be performant and efficient, with a computational cost that scales only linearly with the number of points. Further, three models are introduced that utilize this layer for the generation of jets as point clouds. The models are evaluated on the JetNet [62] datasets that were previously introduced for the comparison of point cloud models in HEP.

Finally, in Chapter 8, the results of this thesis are summarized and an outlook on future research directions is given.

Part I

Theory and Methodology

Chapter 1

Particle Physics

While it is difficult to say when the scientific field of elementary particle physics originated, early important events were the discovery of the electron by J. J. Thompson in 1897 [63] and Rutherford's scattering experiments in 1911 [64]. The latter indicated that nuclei are tiny and that they are separated by a (comparatively) vast space in between them. These discoveries gave rise to Niels Bohr's model of atoms in 1913 [65] involving the discovery of the proton as the nuclei of the hydrogen atom. It was followed up by Chadwick's discovery of the neutron in 1932 [66]. At the time, it seemed that all matter consists of protons, neutrons, and electrons. Yet over the following decades, a more complex picture of the fundamental matter particles emerged. All these advancements since the early days of particle physics are combined in the most complete theory of matter to date: the *Standard Model of particle physics*.

In this chapter, the theoretical foundations of this thesis are laid out. In Section 1.1 an overview of the Standard Model is given and in Section 1.2 the different terms of the Standard Model Lagrangian are discussed. Phenomena currently not explained by the Standard Model are discussed in Section 1.3. Section 1.4 gives an introduction to current and future collider experiments used to probe the Standard Model and search for new physics. Finally, Section 1.5 discusses the importance of particle physics simulations for consolidating theories with experiments and Section 1.6 introduces the concept of jets in high-energy physics.

An in-depth introduction to modern particle physics can be found in References [67–70]. Note that in this thesis, the speed of light c and the reduced Planck constant $\hbar$ are set to unity, i.e. $c = \hbar = 1$, allowing for a customary and simple expression of units.

1.1 The Standard Model of Particle Physics

The Standard Model (SM) of particle physics is widely recognized as one of the most successful theories in physics. It encompasses all known fundamental particles of matter and the interactions ("forces") that act between them. These interactions include the *electromagnetic force*, the *strong force*, and the *weak force* — three out of the four fundamental forces, i.e. *gravity* is not included. The electromagnetic interaction is governed by *Quantum Electrodynamics* (QED) [71] and the strong force by *Quantum Chromodynamics* (QCD) [72].

Table 1.1: The three generations of fermions of the Standard Model and several of their properties. All fermions have an anti-matter counterpart and each quark comes in one of three colors. The masses are taken from Reference [76]. Table adapted from Reference [77].

	Leptons		Quarks	
	charged	neutrino	up-type	down-type
Charge [e]	± 1	0	$+2/3$	$-1/3$
Spin	1/2	1/2	1/2	1/2
Interacts electromagnetically	yes	no	yes	yes
Interacts weakly	yes	yes	yes	yes
Interacts strongly	no	no	yes	yes
1st generation	e (electron)	ν_e	u (up)	d (down)
Mass	~ 511 eV	< 0.8 eV	~ 2.2 MeV	~ 4.7 MeV
2nd generation	μ (muon)	ν_μ	c (charm)	s (strange)
Mass	~ 105 MeV	< 0.19 MeV	~ 1.27 GeV	~ 93 MeV
3rd generation	τ (tau)	ν_τ	t (top)	b (bottom)
Mass	~ 1.78 GeV	< 18.2 MeV	~ 173 GeV	~ 4.18 GeV

Both the electromagnetic and the weak force are unified in the *Glashow-Weinberg-Salam (GWS) electroweak theory* [73, 74]. These theories use *Quantum Field Theory* (QFT) as their mathematical framework and are symmetrical under special gauge groups described by the *Yang-Mills theory* [75]:

$$\mathrm{SU(3)_C} \times \mathrm{SU(2)_L} \times \mathrm{U(1)_Y} \tag{1.1}$$

where the special unitary group $\mathrm{SU(3)_C}$ describes the strong force and $\mathrm{SU(2)_L} \times \mathrm{U(1)_Y}$ describes the electroweak processes.

In QFT, each fundamental particle is expressed as a field ϕ and the interactions between particles are described by the *Lagrangian density* $\mathcal{L}$ (usually just called 'the Lagrangian') as a function of $\phi(x)$ and the derivatives $\partial_\mu \phi(x)$. In the classical version of the Lagrangian approach, the terms are given by $T - V$, i.e. the difference between the kinetic and potential energy. In particle physics, the potential energy parts of the Lagrangian specify the interactions (forces) in the SM theory.

1.1.1 Fundamental Matter Particles

The fundamental particles of matter in the SM are known as *fermions* and are separated into two classes: *leptons* and *quarks*, each with six *flavors* of particles. They all have a spin of 1/2. The fermions with a negative chirality (left-handed) exhibit a weak isospin of $T = 1/2$ and are arranged into left-handed isospin doublets with a third weak isospin component $T_3 = \pm 1/2$. Fermions with a positive chirality (right-handed) form weak isospin singlets with $T = T_3 = 0$. An overview of the known fermions is given in Table 1.1.

Leptons are categorized into three generations, each consisting of a charged lepton and a neutrino. They are arranged into left-handed weak isospin doublets L:

$$L = \begin{pmatrix} \nu_e \\ e \end{pmatrix}_{\mathrm{L}}, \begin{pmatrix} \nu_\mu \\ \mu \end{pmatrix}_{\mathrm{L}}, \begin{pmatrix} \nu_\tau \\ \tau \end{pmatrix}_{\mathrm{L}}. \tag{1.2}$$

Here the L on the left-hand side of the doublet indicates the left-handed chirality. The charged leptons in the bottom row include the electron (e), the muon (μ), and the tau (τ), each carry a charge of $Q = -1e$, where e is the elementary charge of about 1.6×10^{-19} C [78], and a weak isospin of $T_3 = -1/2$. These particles interact electromagnetically and weakly. Their corresponding neutrinos in the upper row, ν_e, ν_μ, and ν_τ, are electrically neutral, almost massless, carry a weak isospin of $T_3 = +1/2$, and hence only interact via the weak force. The right-handed charged leptons form isospin singlets with $T_3 = 0$:

$$e_{\mathrm{R}},\ \mu_{\mathrm{R}},\ \tau_{\mathrm{R}}. \tag{1.3}$$

Neutrinos are assumed to be (nearly) massless and there are no right-handed neutrinos in the SM. Among the leptons, only the electron and the neutrinos are stable, while the muon decays after a mean lifetime of $\sim 2.2 \times 10^{-6}$ s and the tau after $\sim 2.9 \times 10^{-13}$ s into lighter particles.

Quarks interact via the electromagnetic, the weak, as well as the strong forces and are the building blocks of hadrons. They are also separated into three generations, each consisting of an up-type and a down-type quark, and are arranged into left-handed weak isospin doublets $Q_{\mathrm{L}\alpha}$:

$$Q_{\mathrm{L}\alpha} = \begin{pmatrix} u_\alpha \\ d_\alpha \end{pmatrix}_{\mathrm{L}}, \begin{pmatrix} c_\alpha \\ s_\alpha \end{pmatrix}_{\mathrm{L}}, \begin{pmatrix} t_\alpha \\ b_\alpha \end{pmatrix}_{\mathrm{L}}. \tag{1.4}$$

Here the subscript L stands again for the left-handed chirality and the index α denotes the color charge (red, green, or blue). The up-type quarks in the upper row are the up (u), charm (c), and top (t) quarks with a charge of $Q = +\frac{2}{3}e$ and weak isospin with $T_3 = +1/2$, while the down-type quarks in the bottom row are the down (d), strange (s), and bottom (b) quarks with a charge of $Q = -\frac{1}{3}e$ and weak isospin with $T_3 = -1/2$. The right-handed quarks form isospin singlets with $T_3 = 0$:

$$u_{\mathrm{R}\alpha},\ c_{\mathrm{R}\alpha},\ t_{\mathrm{R}\alpha},\ d_{\mathrm{R}\alpha},\ s_{\mathrm{R}\alpha},\ b_{\mathrm{R}\alpha}. \tag{1.5}$$

This separation of fermions in doublets and singlets incorporates the experimentally observed parity violation of the SU(2) interaction into the SM.

Corresponding to each of these particles, there exists an anti-matter counterpart with the same mass but opposite charge, e.g. the anti-electrons, called *positron*, with a charge of $Q = +1e$ and the anti-up-quark $\bar{u}$ with a charge of $Q = -\frac{2}{3}e$. Right-handed anti-fermions are arranged as isospin doublets and left-handed anti-fermions as isospin singlets.

In addition to the charge, quarks also carry a color charge α. Quarks can have one of three color charges, red, green, and blue, while anti-quarks can have the anti-colors anti-red, anti-green, and anti-blue. While the quarks are color triplets, the leptons are color singlets, so no color index is needed. Quarks have never been observed in isolation and are instead

Table 1.2: The fundamental force mediators of the Standard Model and the Higgs boson. The gluons g come in eight variants, each carrying a different mixture of color and anti-color charges. The masses are taken from Reference [76]. Gluons and the photon are assumed to be massless. Table adapted from Reference [77].

Interaction	Particle	Mass	Charge [e]	Spin
Strong	g	0	0	1
Electromagnetic	γ	0	0	1
Weak	$W^{\pm}$	~ 80.38 GeV	± 1	1
	Z^0	~ 91.19 GeV	0	1
Higgs	h	~ 125.25 GeV	0	0

confined into color-neutral multi-quark bound states, called *hadrons*. Hadrons are further divided into *mesons* and *baryons*. Mesons consist of a quark and an anti-quark, while baryons consist of three quarks. Generally, hadrons are unstable except for the proton (uud) (and the neutron (udd), when bound with a proton in a nucleus).

Hence, the matter surrounding us is made up of protons, neutrons and electrons. We can generate fermions of the higher generations in high-energy particle collisions, but these decay back to the first-generation fermions through diverse decay channels. Additionally, neutrinos constantly move through the universe with minimal interaction and cosmic muons are produced by the interaction of cosmic rays (mostly made up of protons and nuclei) with the Earth's atmosphere.

1.1.2 Fundamental Force Mediators and the Higgs Boson

The SM encompasses three of the four fundamental forces of nature and according to QFT each force is associated with one or multiple particles that acts as mediators of the interaction between fermions. These mediators are the *gauge bosons*, fundamental particles with a spin of 1. An overview of some attributes of these bosons is given in Table 1.2. The fourth fundamental force, gravity, is not part of the current SM and is not yet described by a quantum field theory.

Both the electromagnetic and the weak forces act between charged particles, i.e. charged leptons and quarks. The electromagnetic force is mediated by the massless *photon* (γ) and has an infinite reach decaying with the radius r as $1/r$ according to the Coulomb potential. The weak force allows for flavor-changing interactions such as the β-decay. It is mediated by the three massive bosons, the $W^{\pm}$ and Z^0 bosons, which are electrically charged and neutral, respectively. The weak force has a very short range of $\sim 10^{-17}$ m, governed by the Yukawa potential [79].

The strong force acts in the femtometer range ($\sim 10^{-15}$ m) between color-charged particles and is mediated by the massless *gluons* (g). There are a total of eight different gluons, each carrying a different mixture of color and anti-color charges. Gluons are unique particles in that they carry color charge themselves and are therefore self-interacting. This leads to the phenomenon of asymptotic freedom [80, 81], which means that the strong force becomes

comparatively small at very small ranges, i.e. within a meson or baryon. This gluon-gluon self-interaction also relates to the concept of *color confinement*, which means that only color-neutral composite particles exist, i.e. quarks are never observed in isolation and instead form color-neutral hadrons. Due to the asymptotic freedom, the quarks and gluons are considered to move freely within the hadron. When moved further apart, the strong force increases linearly with the distance and the potential of the color field between the quarks increases until it is energetically favorable to create a new quark-antiquark pair. This limits the reach of the strong force and leads to the creation of more color-neutral hadrons, a process called *hadronization*.

Finally, the scalar Higgs boson (h) and the associated *Higgs mechanism* [82–84] explain the masses of the weak gauge bosons $W^{\pm}$ and Z^0 as well as the masses of the electrically charged fermions. For the conservation of local symmetry, the electroweak gauge group $\mathrm{SU(2)_L \times U(1)_Y}$ usually would not include massive fermions and bosons. The Higgs mechanism allows for the spontaneous breaking of the electroweak symmetry and introduces a self-interacting scalar field, which predicts the existence of the Higgs boson. It is the only spin 0 particle in the SM, is neutrally charged, and has a mass of about 125.25 GeV [76]. It was discovered in 2012 by the ATLAS and CMS experiments at the Large Hadron Collider (LHC) [19, 20] and its discovery was awarded the Nobel Prize in Physics in 2013. The Higgs boson couples to all massive particles and can decay into all of them, except into the top quark due to its large mass. Since its discovery, an important research challenge in HEP is the precision measurement of the Higgs boson properties, such as its mass, its decay width and the couplings to other particles, in order to validate the SM and to search for new physics beyond the Standard Model (BSM).

1.2 The Standard Model Lagrangian

The subsequent subsections follow largely the derivation of the Standard Model Lagrangian as introduced in References [68, 70].

The SM Lagrangian of a spin-1/2 fermion (the *Dirac Lagrangian*) is qualitatively motivated to give the experimentally plausible equations of motions. This free particle Lagrangian is given by

$$\mathcal{L}_{\mathrm{Dirac}} = \bar{\psi}(i\gamma^{\mu}\partial_{\mu} - m)\psi \tag{1.6}$$

where ψ is a fermion spinor field, m is the mass of the particle, and γ^{μ} are the Dirac matrices. The first term corresponds to the kinetic energy of the free particle and the second term to the potential.

As any equation written with covariant derivatives is gauge-invariant, replacing the partial derivatives ∂_{μ} with covariant derivatives D_{μ} (i.e. $\bar{\psi}\gamma^{\mu}\partial_{\mu}\psi \rightarrow \bar{\psi}\gamma^{\mu}D_{\mu}\psi$) yields the gauge-invariant interaction Lagrangian. Here the derivative corresponds to one of the three aforementioned local gauge symmetry groups.

Demanding local gauge invariance for the electromagnetic gauge group U(1), the covariant derivative is given by

$$D_{\mu} = \partial_{\mu} - ig_1 A_{\mu} \tag{1.7}$$

where A_μ is the interaction field (i.e. the electromagnetic field) and g_1 is the coupling strength (i.e. the electric charge Q). As the field A_μ is described by a four-vector, it gives rise to a spin-one particle, the photon. More general, the covariant derivative can be written as

$$D_\mu = \partial_\mu - ig_1 \frac{Y}{2} B_\mu \tag{1.8}$$

where Y is the hypercharge generator and B_μ a massive abelian field.

Analogous in the electroweak gauge group SU(2), three fields W_μ^i are introduced to achieve invariance under weak isospin transformations, where $i = [1, 2, 3]$ correspond to the three Pauli matrices σ^i. The covariant derivative is then given by

$$D_\mu = \partial_\mu - ig_2 \frac{\sigma^i}{2} W_\mu^i \tag{1.9}$$

with g_2 as the coupling strength of the interactions. The fields W_μ^i are realized as spin-one particles since they correspond to four-vector transformations under space rotation — just like the photon in the electromagnetic interaction. These particles have positive, negative and neutral electromagnetic charges:

$$\begin{aligned} W^+ &= \frac{1}{\sqrt{2}}(-W^1 + iW^2), \\ W^- &= \frac{1}{\sqrt{2}}(-W^1 - iW^2), \\ W^0 &= W^3. \end{aligned} \tag{1.10}$$

In the internal SU(3) color space of the strong force, eight fields G_μ^a are introduced to achieve invariance under color transformations, where $a = 1, 2, ..., 8$ correspond to the eight Gell-Mann matrices λ^a. The covariant derivative is given by

$$D_\mu = \partial_\mu - ig_3 \frac{\lambda^a}{2} G_\mu^a \tag{1.11}$$

with g_3 as the coupling strength of the strong interaction.

The full covariant derivative of the SM combines the three gauge groups and is given by

$$D_\mu = \partial_\mu - ig_1 \frac{Y}{2} B_\mu - ig_2 \frac{\sigma^i}{2} W_\mu^i - ig_3 \frac{\lambda^a}{2} G_\mu^a. \tag{1.12}$$

While the first two terms are singlets, the third and fourth terms are 2×2 and 3×3 matrices, respectively. This is consistent as the terms act in different internal spaces. By convention, any term in D_μ acting on a different fermion matrix form, i.e. a different internal space, results in zero. This implies, that quarks and leptons act the same under U(1) and SU(2), since the color charge is only considered in SU(3). Equation 1.12 is one of the main equation of the SM as it explains the gauge bosons and their interactions with the fermions. The interaction fields B^μ, W_i^μ, and G_a^μ give rise to one, three, and eight spin-one gauge bosons, respectively, which all have been observed experimentally.

We term the essential fermionic part of the SM Lagrangian as L_{ferm}. It arises when the covariant derivative Eq. 1.12 is inserted into the kinetic energy term of the Dirac Lagrangian

Eq. 1.6 — ignoring for now the mass terms since they are not gauge invariant (to include the mass in the theory, spontaneous symmetry breaking is needed, as further discussed below). The full Lagrangian of the fermions is then given by:

$$\mathcal{L}_{\text{ferm}} = \sum_f \bar{f} i\gamma^\mu D_\mu f \tag{1.13}$$

where we write the sum over fermions as f, i.e. for the first generation $f = L, e_{\text{R}}, Q_{\text{L}}, u_{\text{R}}, d_{\text{R}}$, where L is the first generation lepton doublet and Q_{L} is the first generation quark doublet.

1.2.1 Electroweak Theory

With the electroweak unification, a single gauge theory with the symmetry group $\text{SU}(2)_{\text{L}} \times \text{U}(1)_{\text{Y}}$ is used to describe both the electromagnetic and the weak forces. The unification allows for a joint description of their respective coupling constants. The quantities corresponding to $SU(2)_{\text{L}}$ and $U(1)_{\text{Y}}$ are the *weak isospin* I_W and the *weak hypercharge* Y_{W}, respectively, where the weak hypercharge can be expressed as $Y_{\text{W}} = 2(Q - I_W)$.

We first further develop the theory based on the electroweak unification since both leptons and quarks act the same under the U(1) and SU(2) gauge groups. Therefore, all equations can be written only with leptons and apply analogous to quarks. Adding the color charge in the SU(3) group is discussed in the next Section 1.2.2. We also limit ourselves to discussing the first fermionic generation as the theory applies analogously to the other generations.

Writing out the fermionic Lagrangian for the right- and left-handed leptons in the U(1) space by inserting Equation 1.8 into Equation 1.13 — ignoring the partial derivative ∂_μ since it occurs in every equation and just yields the kinetic term of the fermionic field — gives:

$$\mathcal{L}_{\text{ferm}}(\text{U}(1), \text{leptons}) = \frac{g_1}{2}\left[Y_{\text{L}}(\bar{\nu}_{\text{L}}\gamma^\mu\nu_{\text{L}} + \bar{e}_{\text{L}}\gamma^\mu e_{\text{L}}) + Y_{\text{R}}(\bar{e}_{\text{R}}\gamma^\mu e_{\text{R}})\right] B_\mu \tag{1.14}$$

where Y_{L} and Y_{R} are the hypercharges of left- and right-handed fermions, respectively. The SU(2) fermionic Lagrangian for the leptons is analogously derived from the covariant derivative in Equation 1.9 in the fermionic Lagrangian:

$$\begin{aligned}&\mathcal{L}_{\text{ferm}}(\text{SU}(2), \text{leptons}) \\ &\quad = \frac{g_2}{2}\left[\bar{\nu}_{\text{L}}\gamma^\mu\nu_{\text{L}}W^0_\mu - \sqrt{2}\bar{\nu}_{\text{L}}\gamma^\mu e_{\text{L}}W^+_\mu - \sqrt{2}\bar{e}_{\text{L}}\gamma^\mu\nu_{\text{L}}W^-_\mu - \bar{e}_{\text{L}}\gamma^\mu e_{\text{L}}W^0_\mu\right]\end{aligned} \tag{1.15}$$

where Equation 1.10 was used to express the $W^\pm$ and W^0 fields. Summing up Equation 1.14 and Equation 1.15 describes all possible interactions of the leptons in the SM (except for the Higgs mechanism).

Next, we aim to unify the terms without charge transfer, i.e. the B_μ and W^0_μ terms associated with the $\bar{e}e$ and $\bar{\nu}_{\text{L}}\nu_{\text{L}}$ interactions. Starting the combination with the neutrino terms (the first term in Equation 1.14 and the first term in Equation 1.15), we define the electromagnetic field A_μ

$$A_\mu \propto g_2 B_\mu - g_1 Y_{\text{L}} W^0_\mu \tag{1.16}$$

and a second field Z_μ is defined as

$$Z_\mu \propto g_1 Y_{\text{L}} B_\mu + g_2 W^0_\mu \tag{1.17}$$

orthogonal to A_μ. This way the neutrinos do not interact with the electromagnetic field.

The remaining neutral terms are the electron terms in Equation 1.14 and the last term of Equation 1.15. By normalizing the above Equations 1.16 and 1.17 and inserting them into the electron terms, we can write the electron charge as

$$e = \frac{g_1 g_2}{\sqrt{g_2^2 + g_1^2}} \tag{1.18}$$

when realizing that the hypercharges are $Y_\mathrm{R} = 2Y_\mathrm{L}$ and setting $Y_\mathrm{L} = -1$. Note that from electromagnetism we know that the charge Q is $-e$ for electrons.

This shows that the definition of A_μ is consistent with the electromagnetic current as it facilitates the interaction of electrons, but not neutrinos. The additional novel "neutral current" Z_μ interacts with both electrons and neutrinos.

As both coupling strengths for electromagnetism and the weak interaction g_1 and g_2 are used to define the electron charge in Equation 1.18, we can define the *electroweak mixing angle* Φ_w (also known as the Weinberg angle) as an important parameter in the SM. It is given by

$$\begin{aligned} \sin\Phi_w &= \frac{g_1}{\sqrt{g_1^2 + g_2^2}} \\ \cos\Phi_w &= \frac{g_2}{\sqrt{g_1^2 + g_2^2}} \end{aligned} \tag{1.19}$$

and can be expressed in terms of e via $e = g_1\cos\Phi_w = g_2\sin\Phi_w$. The mixing angle can be experimentally measured and is about $\sin^2\Phi_w \approx 0.23$ [70]. One can further write the charge Q_f of any fermion f in terms of the weak isospin T_3 component and the hypercharge with

$$Q_f = T_3^f + \frac{Y_f}{2}. \tag{1.20}$$

Hence, a unified form of electromagnetism and the weak interaction emerges that includes a massive neutral Z^0-boson that interacts with any fermion f carrying an electric charge Q_f or a non-zero weak isospin T_3^f. As will be discussed below, the mass of the Z^0-boson can be calculated, and it was discovered in 1983 by the UA1 and UA2 experiments at CERN's Super Proton Synchrotron (SPS) with a mass of about 91 GeV proving the consistency of the theoretical approach [85, 86].

To consider the charged current transitions, we revisit the remaining terms in the SU(2) Lagrangian Equation 1.15, i.e. the terms involving both electrons and neutrinos:

$$\begin{aligned} \mathcal{L}_\mathrm{ferm} &= \frac{g_2}{\sqrt{2}}\left[\bar{\nu}_\mathrm{L}\gamma^\mu e_\mathrm{L} W_\mu^+ + \bar{e}_\mathrm{L}\gamma^\mu \nu_\mathrm{L} W_\mu^-\right] \\ &= \frac{g_2}{\sqrt{2}}\left[\bar{\nu}_\mathrm{L}\gamma^\mu e_\mathrm{L} W_\mu^+ + \mathrm{h.c.}\right]. \end{aligned} \tag{1.21}$$

Note that since the two terms are Hermitian conjugates of each other, it is customary to write only one term and add "h.c." to indicate the addition of the Hermitian conjugate. Here the parity violation of the weak interaction is evident, as only left-handed electrons interact with the $W^\pm$ particles. Once the Lagrangian is extended to include quarks, also right-handed terms are added.

The charged spin-1 bosons, W^+ and W^-, were also discovered in 1983 by the UA1 and UA2 experiments at CERN's Sp$\bar{\text{p}}$S with a mass of about 80 GeV [87, 88]. Before discussing the explicit addition of mass terms with the Higgs mechanism, we will first discuss the strong interaction and the full fermionic Lagrangian including quarks.

1.2.2 Quantum Chromodynamics

While the previously derived electroweak unification theory also applies to quarks, the $\lambda^a G^a$ terms in the covariant derivative of the Lagrangian are 3×3 matrices in color space and yield zeros for leptons. For quarks, there is a contribution as they carry a color charge. For a quark q, the QCD contribution to the Lagrangian is given by

$$\mathcal{L}_{\text{QCD}} \sim \frac{g_3}{2} \bar{q}_\alpha \gamma^\mu \lambda^a_{\alpha\beta} G^a_\mu q_\beta \tag{1.22}$$

with color charges $\alpha, \beta \in \{1, 2, 3\}$, i.e. red, green, and blue, allowing for a color charge transition. One can identify G^a as the eight gluons. As their electromagnetic charge is neutral, they do not interact with the electromagnetic field (unlike the W^i of the weak interaction). Their interaction with the quarks is similar to photons, but since the generators λ^a include off-diagonal elements, the two interacting quarks need to have different color charge. The gluon was first experimentally observed in 1979 by the TASSO experiment at the Positron-Electron Tandem Ring Accelerator (PETRA) at DESY [89].

We can now write the full 'fermion gauge boson Lagrangian', which includes all interactions of leptons and quarks with all gauge bosons. The relevant terms of the $\text{U}(1) \times \text{SU}(2) \times \text{SU}(3)$ Lagrangian for the first fermionic generation are given by

$$\begin{aligned}
\mathcal{L} = & \sum_{f=\nu_e,e,u,d} e Q_f (\bar{f} \gamma^\mu f) A^\mu \\
& + \frac{g_2}{\cos\Phi_w} \sum_{f=\nu_e,e,u,d} \left[\bar{f}_{\text{L}} \gamma^\mu f_{\text{L}} (T^3_f - Q_f \sin^2\Phi_w) + \bar{f}_{\text{R}} \gamma^\mu f_{\text{R}} (-Q_f \sin^2\Phi_w) \right] Z^\mu \\
& + \frac{g_2}{\sqrt{2}} \left[(\bar{u}_{\text{L}} \gamma^\mu d_{\text{L}} + \bar{\nu}_{e\text{L}} \gamma^\mu e_{\text{L}}) W^+_\mu + \text{h.c.} \right] \\
& + \frac{g_3}{2} \sum_{q=u,d} \bar{q}_\alpha \gamma^\mu \lambda^a_{\alpha\beta} q_\beta G^a_\mu .
\end{aligned} \tag{1.23}$$

The first term corresponds to the electromagnetic interaction, the second term to the neutral weak interaction, the third term to the charged weak interaction, and the fourth term to the strong interaction. Missing are the mass terms for the charged leptons, quarks and $W^\pm$ and Z^0 gauge bosons, which when naively added would break the gauge invariance of the theory so far derived. For the fermions, one can see this by expanding the mass term $m\bar{\psi}\psi$ with the left-and right-handed projection operators P_{L} and P_{R}:

$$\begin{aligned}
m\bar{\psi}\psi &= m\bar{\psi}(P_{\text{L}} + P_{\text{R}})\psi = m\bar{\psi}P_{\text{L}}P_{\text{L}}\psi + m\bar{\psi}P_{\text{R}}P_{\text{R}}\psi \\
&= m(\bar{\psi}_{\text{R}}\psi_{\text{L}} + \bar{\psi}_{\text{L}}\psi_{\text{R}})
\end{aligned} \tag{1.24}$$

This would imply that $\bar{\psi}_{\text{R}}\psi_{\text{L}}$ and $\bar{\psi}_{\text{R}}\psi_{\text{L}}$ are SU(2) doublets, although we know from experiments that right-handed fermions are singlets. Hence, adding the mass terms here naively to

the Lagrangian would break SU(2) gauge invariance. Similarly, the naive mass terms for the U(1) gauge boson would be given by

$$\frac{1}{2}m_B^2 B^\mu B_\mu. \tag{1.25}$$

However, since the U(1) gauge transformation is given by $B^\mu \to B'^\mu = B^\mu - \partial^\mu \chi$, the mass term would not be gauge invariant:

$$\frac{1}{2}m_B^2 B^\mu B_\mu \to \frac{1}{2}m_B^2 (B^\mu - \partial^\mu \chi)(B_\mu - \partial_\mu \chi) \neq \frac{1}{2}m_B^2 B^\mu B_\mu. \tag{1.26}$$

This argument also holds analogous for the SU(2) gauge bosons $W^\pm$ and Z^0 and QCD. While the photon and gluons are indeed massless, we know from experiments that the $W^\pm$ and Z^0 bosons are massive. To mitigate this shortcoming, a theory known as the Higgs mechanism was developed. The theory was first introduced in 1964 and confirmed with the discovery of the Higgs boson in 2012.

1.2.3 Spontaneous Symmetry Breaking and the Higgs Mechanism

To understand the Higgs mechanism, which incorporates a spin-zero field, called Higgs field, into the SM, we examine first how to add the concept of spontaneous symmetry breaking to the Lagrangian.

To illustrate the mechanism, we consider the following Lagrangian for a basic 1-dimensional scalar field ϕ:

$$\mathcal{L} = T - V = \frac{1}{2}\partial_\mu \phi \partial^\mu \phi - \left(\frac{1}{2}\mu^2 \phi^2 + \frac{1}{4}\lambda \phi^4\right) \tag{1.27}$$

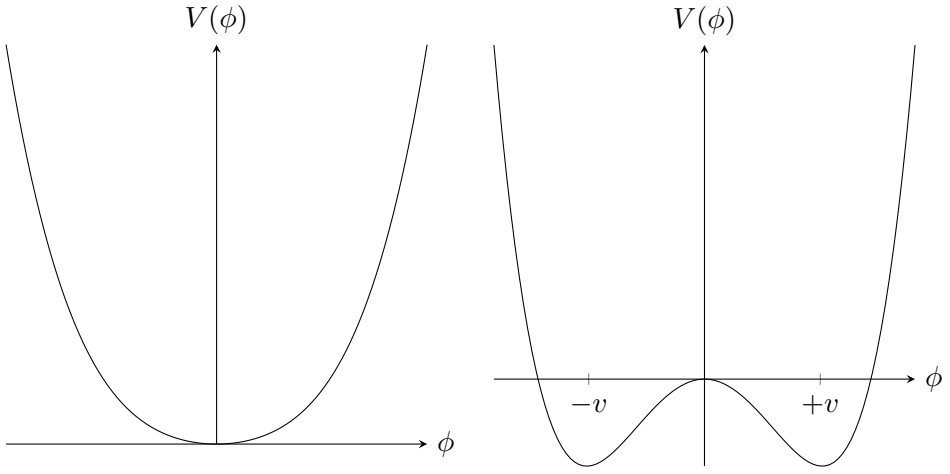
where the last two terms form the potential $V(\phi)$ with the free parameters μ and λ. Importantly, this potential is invariant under $\phi \to -\phi$.

The parameter $\lambda > 0$ is required to have a lower limit of the potential $V(\phi)$ (the ground state in terms of quantum mechanics). As known from perturbation theory in quantum mechanics, one can perturb the system around the ground state to get the excitations. In QFT, the ground state is called the *vacuum* and the excitations are *particles*. We consider two cases of the potential, where μ^2 is either positive or negative. For the case of $\mu^2 > 0$ this results in a single-well potential with a single ground state at $\phi = 0$, for $\mu^2 < 0$ in a double-well potential (also known as a "Mexican hat potential") with two ground states at non-zero *vacuum expectation values (vev)* $\pm v$. A visualization of the resulting potentials for either case is shown in Figure 1.1.

For the simple case of a single-well potential with $\mu^2 > 0$ the vacuum (the minimum) is at $\phi = 0$ and by comparison with the Lagrangian of a simple real scalar field $\phi(x)$ with mass m,

$$\mathcal{L} = \frac{1}{2}\partial_\mu \phi \partial^\mu \phi - \frac{1}{2}m^2 \phi^2, \tag{1.28}$$

it can be seen that, for the single well potential, the parameter μ^2 corresponds to the squared mass.



(a) $\mu^2 > 0$ **(b)** $\mu^2 < 0$

Figure 1.1: One dimensional potential $V(\phi) = \frac{1}{2}\mu^2\phi^2 + \frac{1}{2}\lambda\phi^4$ with $\lambda > 0$ and **(a)** $\mu^2 > 0$ and **(b)** $\mu^2 < 0$. For the second case, the vacuum expectation value v is non-zero, which leads to spontaneous symmetry breaking.

The double-well potential with $\mu^2 < 0$ is more interesting: The minima of the potential are at

$$\phi = \pm v = \pm\sqrt{\frac{-\mu^2}{\lambda}}, \tag{1.29}$$

the two non-zero vacuum expectation values. Here the field ϕ is called a *Higgs field*. To see what particle the Higgs field is giving rise to, the field is perturbed around $\eta = 0$:

$$\phi(x) = v + \eta(x), \tag{1.30}$$

with the choice $\phi = +v$. The opposite choice $\phi = -v$, which would be equally possible since the potential is symmetric under $\phi \to -\phi$.

The following Lagrangian is given near the minimum:

$$\mathcal{L} = \frac{1}{2}(\partial_\mu\eta\partial^\mu\eta) - \left(\lambda v^2\eta^2 + \lambda v\eta^3 + \frac{1}{4}\lambda\eta^4\right) + \text{const.} \tag{1.31}$$

Now the $\lambda v^2\eta^2$ term can once again be interpreted as the mass term for a scalar particle with a mass

$$m_\eta^2 = 2\lambda v^2 = -2\mu^2 \tag{1.32}$$

and the remaining cubic and quartic terms represent self-interactions with the strengths λv and $\frac{1}{4}\lambda$, respectively.

The original Lagrangian in Equation 1.27 is invariant under $\phi \to -\phi$, yet this symmetry is gone in the perturbed Lagrangian in Equation 1.31. The symmetry was broken when the $\phi = +v$ instead of $\phi = -v$ was arbitrarily chosen as the vacuum for the perturbation in Equation 1.30. This phenomenon is called *spontaneous symmetry breaking (SSB)*.

To see how SSB leads to the Higgs mechanism, one needs to consider an equivalent Lagrangian, but with a complex scalar field and consider local gauge invariance (with global

invariance this derivation leads to a massless *Goldstone boson*). The Lagrangian can now be written as

$$\mathcal{L} = (\partial_\mu \phi)^*(\partial^\mu \phi) - \mu^2 \phi^* \phi - \lambda(\phi^* \phi)^2 \tag{1.33}$$

with the complex scalar $\phi = (\phi_1) + i(\phi_2)$ and its complex conjugate $\phi^* = (\phi_1) - i(\phi_2)$. To achieve local gauge invariance, a massless vector field A_μ is introduced and the Lagrangian should be written with covariant derivatives $\partial_\mu \rightarrow D_\mu = \phi_\mu - igA_\mu$. Under the local (hence x-dependent) transformation $\phi(x) \rightarrow \phi'(x) = e^{i\chi(x)}\phi(x)$, the gauge field A_μ has a transformation $A_\mu \rightarrow A'_\mu = A_\mu - \frac{1}{g}\partial_\mu \chi(x)$.

With this local gauge invariance and the covariant derivatives, the Lagrangian can now be written as:

$$\mathcal{L} = (D_\mu \phi)^*(D^\mu \phi) - \mu^2 \phi^* \phi - \lambda(\phi^* \phi)^2 - \frac{1}{4} F_{\mu\nu} F^{\mu\nu} \tag{1.34}$$

with the field strength tensor $F^{\mu\nu} = \partial^\mu A^\nu - \partial^\nu A^\mu$ of U(1) that are used in the kinetic energy terms $-\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$ for the vector field (i.e. the Lagrangian of a free photon field). This term appears since we introduced the gauge field A_μ by moving to the covariant derivative, but it can be ignored for the derivation of the Higgs mechanism. For the first case $\mu^2 > 0$, this Lagrangian describes a charged scalar particle of mass μ interacting with the massless vector field. Again SSB emerges with $\mu^2 < 0$: The Lagrangian describes a multi-dimensional potential with four degrees of freedom, the two real scalars $\phi_{1,2}$ and the two polarization states of A_μ. Analogous as before we can apply a local perturbation with a real scalar h around the vacuum v and write the local scalar field as:

$$\phi(x) = \frac{v + h(x)}{\sqrt{2}}. \tag{1.35}$$

Near the vacuum this leads to the following Lagrangian:

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}(\partial_\mu h)(\partial^\mu h) + \frac{1}{2} g^2 v^2 A_\mu A^\mu - \lambda v^2 h^2 - \lambda v h^3 \\ & - \frac{1}{4}\lambda h^4 + g^2 v h A_\mu A^\mu + \frac{1}{2} g^2 h^2 A_\mu A^\mu - \frac{1}{4} F_{\mu\nu} F^{\mu\nu}. \end{aligned} \tag{1.36}$$

Here the second and third terms describe the mass terms for the vector field A_μ and the scalar field h, respectively. The mass of the vector boson is given by

$$m_A = gv \tag{1.37}$$

and the scalar h is now the real Higgs boson with a mass

$$m_h = \sqrt{2\lambda v^2}. \tag{1.38}$$

The remaining terms describe cubic and quartic self-interactions of the Higgs and interactions with the gauge field A_μ. Overall the theory is still gauge-invariant since the original Lagrangian was gauge-invariant and the perturbation is based on local gauge transformations. This mechanism by which the mass of the gauge boson and a massive Higgs boson emerge is known as the *Higgs mechanism*, and we shall see how it is applied to the Standard Model in the next section.

1.2.4 Higgs Mechanism in the Standard Model

To introduce the Higgs mechanism in the SM, the scalar field (Higgs field) becomes a SU(2) doublet

$$\phi = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} \tag{1.39}$$

with the complex fields

$$\phi^+ = \frac{\phi_1 + i\phi_2}{\sqrt{2}}, \qquad \phi^0 = \frac{\phi_3 + i\phi_4}{\sqrt{2}}. \tag{1.40}$$

These two rotation states are necessary since the Higgs mechanism is used to explain the mass of the neutral and charged gauge bosons (both positively and negatively charged, since $(\phi^+)^* = \phi^-$). The Lagrangian has now an analogous form to Equation 1.33:

$$\mathcal{L}_\phi = (\partial\mu\phi)^\dagger(\partial^\mu\phi) - \mu^2\phi^\dagger\phi - \lambda(\phi^\dagger\phi)^2 \tag{1.41}$$

with the Higgs potential being $V(\phi) = \mu^2\phi^\dagger\phi + \lambda(\phi^\dagger\phi)^2$. As before, the potential is invariant under the local gauge transformation

$$\phi(x) \to \phi'(x) = e^{i\vec{\alpha}(x)\cdot\vec{\sigma}/2}\phi(x) \tag{1.42}$$

where σ^i are the Pauli matrices and $\vec{\alpha}(x)$ are free parameters. Setting $\mu^2 < 0$ in the Higgs potential, we find the minima at

$$\phi^\dagger\phi = \frac{-\mu^2}{2\lambda} = \frac{v^2}{2}. \tag{1.43}$$

A direction in SU(2) space needs to be chosen to perform the perturbation around the vacuum. Here a suitable, yet arbitrary choice for the vacuum is

$$\phi_0 = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ v \end{pmatrix} \tag{1.44}$$

implying that $\phi_1 = \phi_2 = \phi_4 = 0$ and $\phi_3 = v$. The expansion around the vacuum can be performed by the perturbation

$$\phi(x) = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ v + h(x) \end{pmatrix}. \tag{1.45}$$

With this choice, three fields are "gauged away" which implies that there are three global broken symmetries that correspond to the longitudinal polarization state of the three $W^\pm$ and Z^0 bosons. Additionally, the massive Higgs boson emerges once again.

Now the partial derivatives in the Lagrangian Equation 1.41 can be replaced with covariant derivatives to determine the mass and interaction terms. Recall from Equation 1.12, that for the electroweak $\mathrm{SU(2)_L} \times \mathrm{U(1)_Y}$ local gauge symmetry group, the covariant derivative is given by

$$D_\mu = \partial_\mu - ig_1\frac{Y}{2}B_\mu - ig_2\frac{\vec{\sigma}}{2}\vec{W}_\mu. \tag{1.46}$$

Inserting these terms into the Lagrangian and setting $Y = 1$ (which is the hypercharge of the Higgs doublet according to Equation 1.20, since the doublets' lower component is neutral

(see Equation 1.39) and its third weak isospin component $T_3 = -\frac{1}{2}$) results in the following contributions to the Lagrangian that can be interpreted as mass terms:

$$\frac{1}{8}v^2 g_2^2((W_\mu^1)^2 + (W_\mu^2)^2) + \frac{1}{8}v^2(g_1 B_\mu - g_2 W_\mu^3)^2. \tag{1.47}$$

Now one can leverage the insights gained previously from studying the electroweak theory. Using Equation 1.10 the first term equals

$$\left(\frac{1}{2}vg_2\right)^2 (W_\mu^+ W_\mu^-) \tag{1.48}$$

hence, the Higgs mechanism gives the $W^\pm$ bosons a mass of

$$m_{W^\pm} = \frac{1}{2}vg_2. \tag{1.49}$$

Rewriting the second term is slightly more complex but from Equation 1.16 and Equation 1.17 the relation of B_μ and W_μ^3 with the photon field A_μ and the Z^μ are known. With some additional computation, this leads to the mass terms

$$m_{Z^0} = \frac{1}{2}v\sqrt{g_1^2 + g_2^2}, \qquad m_\gamma = 0 \tag{1.50}$$

for the Z^0 boson and the photon, respectively. As expected, the photon turns out massless. Further, using Equation 1.19 the mass ratio of the $W^\pm$ and Z^0 bosons can be written in terms of the electroweak mixing angle Φ_w:

$$\frac{m_{W^\pm}}{m_{Z^0}} = \cos\Phi_w. \tag{1.51}$$

This ratio is a useful quantity to check the soundness of the Standard Model as one can write $\rho = m_W^\pm/(m_{Z^0}\cos\Phi_w)$ which has to equal unity. The observed gauge boson masses and the mixing angle satisfy this condition to an accuracy of about 0.1%, yet deviations that could be observed in the future might hint at physics beyond the Standard Model (further discussed in the next Section 1.3).

1.2.5 Higgs Mechanism for Fermionic Masses

The Higgs mechanism is also able to explain the masses of the fermions. To show this one can add lepton interaction terms to the Lagrangian:

$$\mathcal{L}_{\text{int}} = g_e(\bar{L}\phi e_{\text{R}} + \phi^\dagger \bar{e}_{\text{R}} L) = g_e(\bar{L}\phi e_{\text{R}} + \text{h.c.}) \tag{1.52}$$

where L is the lepton doublet of the first generation, ϕ is the Higgs doublet, and g_e is the Yukawa coupling constant of the electron to the Higgs field. This Lagrangian can be expanded around the vacuum with the previous perturbation in Equation 1.45 yielding

$$\mathcal{L}_{\text{int}} = \frac{g_e v}{\sqrt{2}}(\bar{e}_{\text{L}} e_{\text{R}} + \bar{e}_{\text{R}} e_{\text{L}}) + \frac{g_e}{\sqrt{2}}(\bar{e}_{\text{L}} e_{\text{R}} + \bar{e}_{\text{R}} e_{\text{L}})h. \tag{1.53}$$

The first term can be interpreted as the electron mass given by

$$m_e = \frac{g_e v}{\sqrt{2}}, \tag{1.54}$$

so there is finally a term in the SM Lagrangian that achieves non-zero fermion masses as known from experiments. The second term describes the electron-Higgs vertex with a strength of $g_e/\sqrt{2} = m_e/v$, i.e. the probability of a Higgs decaying into e^+e^- or for an electron/positron to radiate a Higgs boson. As no terms for the neutrinos appear, they still do not acquire a mass through the Higgs mechanism (a shortcoming of the SM, see Section 1.3).

To deduce the mass terms of the first-generation quarks analogously, the conjugate of the Higgs field doublet is written as

$$\phi_c = i\sigma_2\phi^* = \begin{pmatrix} \phi^{0*} \\ -\phi^- \end{pmatrix} \tag{1.55}$$

with σ_2 being the second Pauli matrix. This conjugate has the opposite hypercharge $Y = -1$. For a perturbation around the vacuum one finds

$$\phi_c = \frac{1}{\sqrt{2}} \begin{pmatrix} v + h \\ 0 \end{pmatrix}. \tag{1.56}$$

The interaction Lagrangian for the first quark generation is

$$\mathcal{L}_{\text{int}} = g_d \bar{Q}_{\text{L}} \phi d_{\text{R}} + g_u \bar{Q}_{\text{L}} \phi_c u_{\text{R}} + \text{h.c.} \tag{1.57}$$

where Q_{L} is the quark doublet of the first generation, d_{R} and u_{R} are the right-handed down and up quarks, and g_d and g_u are the Yukawa coupling constants of the down and up quarks to the Higgs field. Expanding the Higgs field and its conjugate around the vacuum, one finds the mass terms for the down and up quarks similarly to the electron mass:

$$m_d = \frac{g_d v}{\sqrt{2}}, \qquad m_u = \frac{g_u v}{\sqrt{2}}. \tag{1.58}$$

The couplings of the quarks to the Higgs field are derived analogously. This process can be repeated for the second and third fermion generations. With the Higgs mechanism, we are hence able to explain the masses of the fermions in the Standard Model. As their couplings to the Higgs field are proportional to the masses, heavy fermions couple stronger to the Higgs boson and therefore high energies are needed to produce these fermions for studying Higgs physics.

This concludes the discussion of the SM Lagrangian, which is the centerpiece of the SM and describes the interactions of all known elementary particles. It encompasses the knowledge gained by decades of analyzing the data taken at various particle physics experiments, especially at colliders. A large focus of the current particle physics research lies in the precision measurements of all the free parameters of the SM — together with direct searches for new particles. It is well known that the current SM is not the full story of how the universe works and its limitations are well described. Precision measurements at current and future colliders and alternative experiments such as astrophysical observations will help us shape the next iteration of particle physics theory. Next, a few of these limitations are outlined.

1.3 Physics Beyond the Standard Model

While the SM is a very successful theory, it does not explain all observed phenomena in the universe. It is known to be incomplete and does not constitute a complete *theory of*

everything. Addressing the shortcomings of the SM is one of the main goals of high-energy physics, and is done by searching for *physics beyond the Standard Model* (BSM), i.e. for new particles, forces, or interactions that are not described by the SM. Among the experimental results currently not explained by the SM are the following phenomena:

- **Gravity:** As one of the fundamental forces of nature, gravity is famously not included in the SM. On macroscopic scales, the influence of gravity is clearly visible and well described by the theory of general relativity [90]. A major advancement of physics would be the unification of the SM with general relativity.

- **Dark matter:** Observed gravitational effects such as galaxy (cluster) rotations indicate that of the total mass-energy budget of the universe, only about 5% is made up of ordinary baryonic matter and about 25% consists of so-called dark matter, which appears to only interact gravitationally and does not emit or absorb light. The remaining mass-energy does not clump via gravitation and is attributed to dark energy, generally considered a constant energy density in the general relativity equations. Considering that the SM does not explain 95% of the universe's mass-energy content is a very strong motivation for BSM searches. It could be that dark matter is made up of new particles that interact very weakly with the SM. Such theoretical particles could be discovered at high-energy colliders and are generally termed as *weakly interacting massive particles* (WIMPs).

- **Neutrino masses:** The SM does not include a process that would allow for neutrinos to have mass. However, the phenomenon of neutrino oscillations in which neutrinos change flavor as they propagate through space has been observed and requires neutrinos to have mass. Such neutrino oscillations are confirmed by various experiments, i.e. by the Super-Kamiokande experiment [91] and the Sudbury Neutrino Observatory [92] awarded with the 2015 Nobel Prize for Physics. Recently, an upper limit on the mass of the electron antineutrino of $m_\nu < 0.45$ eV at 90 % confidence level was set by the KATRIN experiment [93].

- **Matter-antimatter asymmetry:** A widely accepted theory on the inception of the universe is the Big Bang, which led to the creation of all matter. However, the SM does not explain the observable asymmetry of baryonic to anti-baryonic matter.

Further, the SM has multiple conceptual shortcomings, including the following issues:

- **Hierarchy problem:** Unless fine-tuned quantum corrections are applied when calculating the Higgs boson mass in a QFT, its mass would raise to the Planck scale at $\sim 10^{18}$ GeV. This would also raise the masses of all quarks, charged leptons and $W^\pm$ and Z^0 bosons. It is unclear why the Higgs boson mass is at the electroweak scale of ~ 100 GeV and such fine-tuned corrections are considered *unnatural*, according to the theory of *naturalness* [94]. The theory of supersymmetry [95] would avoid this fine-tuning since bosonic and fermionic quantum corrections have opposite signs and would cancel each other out, yet no experimental evidence for supersymmetric particles has been found so far.

- **Force unification:** With the electroweak theory, electromagnetism and the weak interaction are unified. This unification suggests that an additional unification of the electroweak theory with the strong interaction should be possible. Such a theory is referred to as a *Grand Unified Theory* (GUT), which would also include the unification of quarks and leptons, i.e. in an SU(5) representation [70].
- **Ad-hoc parameters:** The SM requires 19 numerical constants (not counting some associated with Neutrino masses) that are not predicted by the theory and need to be measured experimentally. These include the masses of fermions and the Higgs boson, CKM mixing angles, and coupling constants for each gauge group. Additionally, it is unclear why there are exactly three lepton and quark generations.

Overall, BSM physics is a major field of research in high-energy physics and is pursued with many experiments, including collider experiments, neutrino experiments, and astrophysical observations. The work in this thesis focuses on analysis methods for collider experiments, hence we will discuss next how to probe the SM at high-energy colliders.

1.4 Collider Experiments

Over the recent decades, experiments at particle accelerators have been very successful in studying nature at the smallest scales. To produce and study elementary particles, a beam of particles such as electrons or protons is accelerated to high energies and is either collided with a fixed target (*fixed target experiment*) or another beam of particles (*collider experiment*). All the heavy elementary particles and gauge bosons up to the discovery of the Higgs boson in 2012 were discovered at collider experiments. Highly energetic beams are needed as the mass of the produced particles during the collision is limited by the center-of-mass energy $\sqrt{s}$. With this large success of collider experiments, it is unsurprising that over the past decades, multiple colliders with ever-increasing center-of-mass energies were built. A list of several colliders built since 1961 is shown in Table 1.3. A detailed introduction into accelerator physics and design can be found in Reference [96].

In general, colliders are either built as circular or as linear accelerators. Most colliders so far were built as circular synchrotrons, with the exception of the Stanford Linear Collider (SLC). The advantage of circular colliders is that the particles can be accelerated over many revolutions in the same storage ring to achieve very high collision energies. However, this energy gain is countered by *synchrotron radiation*. Whenever a charged particle alters its direction energy is lost in the form of emitted photons. The energy loss ΔE through synchrotron radiation is determined by

$$\Delta E \propto \frac{E^4}{m_0^4 \cdot r} \tag{1.59}$$

where E is the energy of the particle, m_0 its rest mass, and r is the radius of the circular accelerator. We see that the energy loss is inversely proportional to the particle mass and the machine radius. For a fixed radius and fixed energy gain with accelerator cavities, light particles such as electrons loose more energy than heavier particles like protons. Hence, the

Table 1.3: An abridged list of notable particle colliders. With the exception of the SLC, all of them are circular accelerators. The size indicates the length of the beam pipe.

Type	Name	Operation	Size	max. $\sqrt{s}$	Comment
e^+e^-	Anello Di Accumulazione (ADA)	1961—1964	3 m	500 MeV	First e^+e^- collider
	Positron-Electron Tandem Ring Accelerator (PETRA)	1978—1986	2 km	46 GeV	Gluon discovery
	Stanford Linear Collider (SLC)	1989—1998	4 km	90 GeV	First linear collider
	Large Electron-Positron Collider (LEP)	1989—2000	26.7 km	209 GeV	Largest e^+e^- collider to date
e^-p	Hadron–Electron Ring Accelerator (HERA)	1992—2007	6.3 km	320 GeV	Only lepton-hadron collider to date
pp	Intersecting Storage Rings (ISR)	1971—1984	940 m	62 GeV	First hadron collider
$p\bar{p}$	Super Proton–Antiproton Synchrotron (Sp$\bar{\text{p}}$S)	1981—1991	6.9 km	900 GeV	First $p\bar{p}$ collider Discovery of $W^\pm$ and Z^0
$p\bar{p}$	Tevatron	1987—2011	6.3 km	1.96 TeV	Top quark discovery
pp	Large Hadron Collider (LHC)	2010—present	26.7 km	13.6 TeV	Higgs boson discovery

easiest way to achieve higher energies when accelerating a given particle is to build a larger circular collider, explaining why in Table 1.3 we see increasing accelerator circumference as time progresses. An exception is the SLC, which was built as a linear collider. For linear colliders, synchrotron radiation can be neglected allowing for potentially larger achieved energies for light particle such as electrons. Higher energies can also be achieved by increasing the length of the acceleration distance and by improving the accelerator technology.

We further differentiate colliders by the type of particles they accelerate, either leptons or hadrons. Colliding stable leptons, i.e. electrons and positrons, has the advantage that the initial state of the system, including the center-of-mass energy and the spin orientation (for polarized beams) is well-defined. This enables precision measurements with little background. When colliding hadrons, in particular (anti-)protons, this initial state is not well-defined, as hadrons are composite particles consisting of quarks and gluons. Since not the whole protons are colliding the actual center-of-mass energy is uncertain. Further, the strong interaction between the quarks and gluons leads to a large QCD background. For hadron collisions, an important quantity is the momentum transverse to the beam-axis – denoted p_T– as in the initial state it is zero and therefore well-defined.

However, an advantage of colliding hadrons is that they are significantly heavier than electrons allowing for potentially much higher maximum center-of-mass energies. Therefore, hadron colliders are often dubbed “discovery machines” as they enable us to push the energy frontier and discover heavier particles than might be possible with a lepton collider, which on the other hand is excellent for precision measurements at comparatively lower energies. Currently, the Large Hadron Collider (LHC) is operated as the largest discovery machine ever built.

1.4.1 The Large Hadron Collider (LHC)

The LHC [97] is the most powerful particle collider ever built. It reuses the 26.7 km circumference tunnel infrastructure of the Large Electron-Positron Collider (LEP) and started its first data-taking run in 2010. Since then, it was upgraded and currently achieves an energy of $\sqrt{s} = 13.6$ GeV when colliding protons. Apart from protons, also heavy ions, i.e. lead nuclei, can be accelerated and collided. The LHC not only achieved the highest center-of-mass energies ever, but it also is the collider with the highest (instantaneous) luminosity. The instantaneous luminosity in a circular particle collider (assuming equal beam properties) is defined as

$$L = \frac{f^2 n_p^2 n_b}{A} \tag{1.60}$$

where f is the revolution frequency of the beams, n_p is the number of particles per bunch, n_b is the number of bunches per beam, and A is the cross-sectional area of the beams. Luminosity has the units of inverse area per time, i.e. $\mathrm{cm}^{-2}\mathrm{s}^{-1}$, and it determines how many collisions occur in a certain time. Together with the cross-section σ for a certain kind of event, the number of events of a that type that are produced over a given time between T_1 and T_2 can be calculated as:

$$N = \sigma \int_{T_1}^{T_2} L dt \tag{1.61}$$

i.e. it depends on the *integrated luminosity* $\int L dt$. Together with the energy, the luminosity is a crucial parameter of any collider. Since the cross-section is inversely proportional to the square of the mass scale the experiments are probing, a very high luminosity is needed to produce a sufficient amount of events at the energy frontier.

The LHC is designed for a luminosity of $L = 1 \times 10^{34}$ $\mathrm{cm}^{-2}\mathrm{s}^{-1}$ and in the year 2018 achieved a peak luminosity of $L = 2 \times 10^{34}$ $\mathrm{cm}^{-2}\mathrm{s}^{-1}$. To achieve this luminosity, the LHC was designed for $n_p = 1.15 \times 10^{11}$ protons per bunch, $n_b = 2808$ bunches per beam, and a bunch crossing frequency of 40 MHz, which translates to a bunch spacing of 25 ns [97]. Such a high luminosity is needed to produce about one Higgs boson every second. It is expected, that by the end of the LHC Run 3 (at the end of 2024), the LHC will have achieved a total integrated luminosity of about 350 fb^{-1}[1] surpassing the original design goal of 300 fb^{-1} [21].

To exploit the LHC further and to probe the energy frontier more effectively by producing more rare events, a major upgrade – the high-luminosity LHC (HL-LHC) [21] —is planned. The HL-LHC will increase the collision rate by a factor of 5 and the integrated luminosity by about a factor of 10 compared to the LHCs' original design — achieving a peak instantaneous luminosity of $L = 5 \times 10^{34}$ $\mathrm{cm}^{-2}\mathrm{s}^{-1}$ and over 12 years an integrated luminosity of 3 ab^{-1}. For the upgrade several accelerator components are improved, including new magnet designs, stronger 11-12 T superconducting magnets, more compact superconducting RF cavities for precise phase control of the beams, and an overall remodeled beam collimation system. The goals are to increase the bunch intensity to 2.2×10^{11} protons per bunch, to direct and collimate the bunches for an increased overlapping area, and to optimize the operation by luminosity leveling. The luminosity leveling is needed to mitigate extreme *pile-up* in the experiments, i.e. to limit the additional proton-proton interactions that confuse the study of

[1] For reference: 1 $\mathrm{b}^{-1} = 10^{24}$ cm^{-2} and 1 $\mathrm{fb}^{-1} = 10^{39}$ cm^{-2}

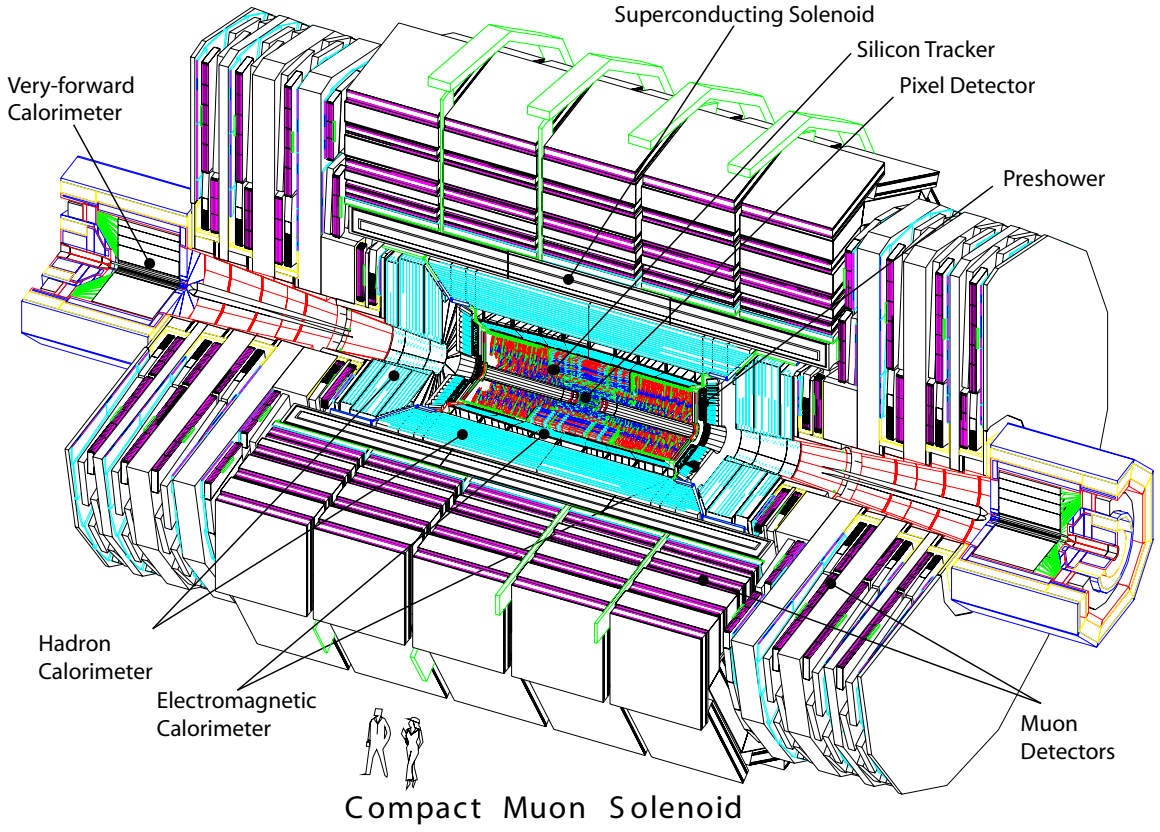


Figure 1.2: Cutaway overview of the CMS detector at the LHC. It is 21.6 m long and has a diameter of 14.6 m. Figure taken from Reference [98].

the primary interaction vertex of interest. While currently about 30–60 pile-up interactions occur during a collision, with the HL-LHC, on average of 140–200 are expected. This necessitates detectors with better separation of charged and neutral particles (see Chapter 2) as well as sophisticated algorithms to filter out the pile-up.

Currently, it is envisioned that the HL-LHC will be operational in 2029. With this significant increase in collisions, also the experiments' data collection pipelines will need to be upgraded, and computing systems will need to account for increased computational cost for the reconstruction of events. For successful physics analyses the amount of simulated collisions needs to scale with the experimentally recorded ones, leading also to increases in cost for simulation of data. While the technological challenges to upgrade the LHC are largely solved, more innovation is needed for simulations to be computationally efficient enough to fit in the currently envisioned computing budget (see also Section 1.5). This thesis aims to tackle this challenge with the results presented in the Chapter 5, 6, and 7.

1.4.2 LHC Experiments

The beams of the LHC are brought to collision at four interaction points where large-scale particle detectors are set up to record the particles produced during the collisions. These

include the two general-purpose detector experiments ATLAS (A Toroidal LHC ApparatuS) and CMS (Compact Muon Solenoid) as well as the ALICE (A Large Ion Collider Experiment) detector for heavy-ion collisions, and the LHCb (Large Hadron Collider beauty) detector purposely built to study B mesons. So far the greatest scientific success of the LHC experiments was the discovery of the Higgs boson at a mass of about 125 GeV in 2012 by both the CMS and ATLAS experiments [19, 20].

General-purpose detectors like ATLAS and CMS consist of a strong magnet and several sub-detectors for dedicated measurements that encapsulate the interaction point in a solid angle of 4π with multiple layers of components. As an example, a cutaway overview of the CMS detector is shown in Figure 1.2. Similar to other general-purpose detectors, this detector consists of four main parts [98]:

1. **Magnet:** An important part of the detector is the strong solenoid magnet that creates a magnetic field parallel to the beam axis. Charged particles created by the collision are bent in the magnetic field allowing the determination of the sign of their charge as well as their momentum perpendicular to the beam axis. In CMS the magnet is made up of cylindrical coils of superconducting fiber which create a magnetic field of 3.8 T. A steel yoke is used to limit the reach of the magnetic field.

2. **Tracker:** Closest to the beam pipe is the *tracking system*, which measures the trajectories of charged particles produced by the collision. Due to the trajectories being bent in the magnetic field, the sign of their charge and their momentum perpendicular to magnet lines (or beam line), i.e. p_{T}, can be calculated. The reconstructed particle tracks are further used to reconstruct and identify the primary leading vertex with the highest total energy, secondary decay vertices, and secondary proton-proton interactions, which can be used to mitigate pile-up.

   The Tracking system at CMS uses technologies such as silicon semiconductor-based pixel and microstrip detection elements. These operate on the principle that when a charged particle traverses the silicon, an electron-hole pair is created resulting in a small charge that is measured. In total, it is 5.8 m long with a diameter of 2.5 m and covers a pseudorapidity of $|\eta| < 2.5$. The first of multiple layers starts at only 2.9 cm away from the beam pipe and therefore needs to be very resistant to radiation. After an upgrade in 2017, the pixel tracker has 124 million pixels each with a size of 100×150 $\mu\mathrm{m}^2$ achieving a precision of 10 μm when determining the particle's origin [99].

3. **Calorimetry:** The *calorimeter system* that surrounds the tracking system consists of dense material to fully absorb particles and to measure the deposited energy. While tracking systems are optimized to be particularly sensitive to charged particles, calorimeters are used to detect both charged and neutral particles (except neutrinos). They are usually separated into an inner electromagnetic calorimeter (ECAL) and an outer hadronic calorimeter (HCAL), each with different material compositions to best measure the respective particle types.

   The CMS ECAL is a homogenous calorimeter using a total of 75,848 lead tungstate ($PbWO_4$) crystal scintillators and is divided up into a barrel and endcap region. In the

barrel, the 61,200 crystals are 230 mm long which equates to 25.8 radiation lengths (X_0) and have a cross-section of 22×22 mm^2 at the front facing the beam pipe. It extends to $r = 1.77$ m away from the beam pipe. The 17,648 endcap crystals are similarly 220 mm long (24.7 X_0) with a front area of 28.62×28.62 mm^2. In front of the endcaps, a preshower detector is placed. It is a two-layer sampling calorimeter using lead absorbers and silicon strip sensors with the main usage of identifying neutral pions.

Enclosing the ECAL, the CMS HCAL is a sampling calorimeter using passive steel or brass absorbers with various thicknesses and 70,000 active plastic scintillator tiles. The light from the scintillators is picked up by wavelength-shifting fibers. The barrel section extends to $r = 2.95$ m and consists of 17 active layers with a thickness of 3.7 mm. The first and last absorber layers are steel while the remaining passive layers are made of brass. This results in an absorber thickness of 5.82 interaction lengths (λ_{int}) at $\eta = 0$ and 10.6 λ_{int} at $|\eta| = 1.3$. The endcap section covers a region of $1.3 < |\eta| < 3.0$ and is made up of 19 active and passive brass layers.

Both the tracking system and the calorimeters are placed inside the solenoid magnet. In addition, there is a small outer section of the barrel HCAL (a "tail catcher") as well as a forward HCAL section placed 11.2 m away from the interaction point. A detailed overview of calorimetry is given in Chapter 2.

4. **Muon System:** The outermost part of the detector is inhabited by the *muon detector*, which is designed to detect muons since they traverse the tracker and calorimeter systems without much energy loss. It acts essentially as an extension of the tracker system, by tracking the bent trajectory of traversing muons. Together with the information from the tracking detector, the muon momenta can be accurately calculated. At CMS, the muon system has the largest volume of active material and consists of about 1,472 chambers including 250 drift tubes, 540 cathode strip chambers, 610 resistive plate chambers, and 72 gas electron multiplier chambers. The chambers are organized into four stations, which are interlaced with steel plates doubling as the aforementioned flux return yoke for the magnet and as a hadron absorber. For the physics program of CMS, the muon system is of particular importance since it allows a good measurement of the clean Higgs signature via its decay into four muons $h \to Z^0(\to \mu^-\mu^+)Z^0(\to \mu^-\mu^+)$.

The coordinate system used by the CMS experiment is similar to that employed by other detectors and uses a right-handed system with the origin at the collision point inside the center of the detector. The x-axis points towards the center of the collider ring, the y-axis upwards towards the surface, and the z-axis along the beam pipe in the counterclockwise direction. When speaking of spherical coordinates in the detector context, r describes the radial distance from the z-axis, ϕ is the azimuthal angle in the $x - y$-plane, and θ is the polar angle measured from the z-axis. The pseudorapidity depends on the polar angle as $\eta = -\ln(\tan(\theta/2))$.

At the nominal LHC instantaneous luminosity of $L = 1 \times 10^{34}$ cm^{-2} s^{-1} with a bunch crossing every 25 ns (40 MHz), about 25 proton-proton interactions occur during a single bunch crossing. The detector response that is recorded as a result of a bunch crossing is called an *event*. For the center-of-mass energy of 13 TeV achieved by the LHC, the total cross-section

for all collisions is $\sigma_{\text{TOT}} \approx 100$ mb translating to about $\mathcal{O}(10^9)$ collisions per second with the nominal luminosity. It is very difficult to filter out the physically "interesting" events which typically have a cross-section of 0.1–1 pb, i.e. only 10^{-11} of σ_{TOT} [70]. Additionally, it is technically impossible to process and store the data measured from every single event, therefore a *trigger system* is used to filter and significantly reduce the amount of data.

The trigger system usually have multiple levels, first hardware-based ones followed by software-based triggers, i.e. ATLAS uses a three-stage trigger while CMS uses a two-stage trigger combining a hardware-based Level-1 trigger and a software-based High-Level Trigger (HLT) [100]. The Level-1 trigger reduces the event rate to 100 kHz by selecting events with interesting features such as ionization deposits consistent with muons. The software-based HLT reduces the event rate further to about 2–3 kHz by reconstructing objects such as electrons, muons and jets and using those objects to select interesting events. The selected events, for which all detector systems are read-out, are then stored for later offline analysis. For reference, the event size is about 1 megabyte (depending on the number of particles produced in the collision), so each second about 2–3 gigabytes of events are stored resulting in multiple petabytes of data acquired over a year by each experiment. Capturing interesting events without discarding major discoveries requires careful tuning of the trigger system. This challenge becomes even greater when moving towards the HL-LHC with a five times higher luminosity.

With the HL-LHC come new challenges for the detector experiments: The higher instantaneous luminosity means more pile-up, i.e. more interactions the detector needs to resolve and to record, and the higher integrated luminosity means that the detector sustains more radiation damage requiring stronger materials. Therefore, for the HL-LHC the LHC detector experiments are going to be upgraded [101]. This includes replacing parts of the sub-detectors and much of the electronics to make the detector able to withstand the increased radiation. To cope with the higher *in-time pile-up* (IT), the granularity of the tracker and parts of the calorimeter will be increased. This allows for better segmentation of the many more trajectories that originate from the interaction region and enables the application of precise particle-flow algorithms (PFA) (see Section 2.3). Additionally, *out-of-time pile-up* (OOT) needs to be suppressed, i.e. energy depositions in the calorimeter that leak into an event from a previous or later bunch crossing. OOT can be mitigated by increasing the time resolution of the calorimeter cells.

To achieve a higher detector granularity in both space and time, mainly the tracker and the endcap calorimeters will be changed during the "Phase II" upgrade of the CMS detector. For example, the inner tracker pixel system will be replaced to feature two billion pixels (up from 124 million pixels currently). The endcap calorimeters will be replaced with a high-granularity calorimeter (HGCAL) [22] which will have 6.4 million read-out channels using hexagonal silicon cells (with a size of 0.5 or 1 cm^2) and scintillator tiles (4 to 32 cm^2) read out by silicon photomultipliers ("SiPM-on-tile design"). Just like the current endcap calorimeter, it will be separated into an ECAL and an HCAL region. The ECAL will consists of 26 layers using silicon sensors with passive copper, copper-tungsten and lead absorbers. The HCAL will be made of 21 active silicon and scintillator layers interleaved with steel absorbers. In addition, new systems are added to the detector such as a minimum-ionizing

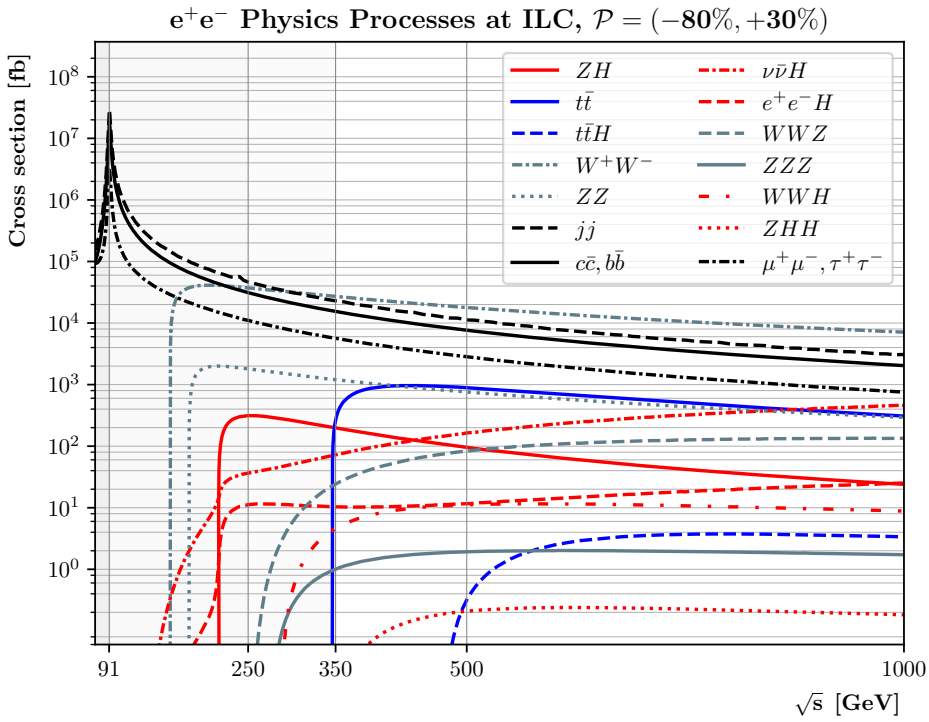


Figure 1.3: Cross-sections of various physics processes at an e^+e^- collider as functions of the center-of-mass energy $\sqrt{s}$. Cross-sections were calculated for the International Linear Collider (ILC) with predominantly left-handed beam polarization (-80%/ for e^+ and +30% for e^+). Figure taken from Reference [102].

particle precision timing detector (MTD), a luminosity detector, and a gas electron multiplier (GEM) detector for muon detection in the very-forward region.

1.4.3 Future Collider Experiments

While the physics program around the HL-LHC is decided upon and will last until the mid-2030s, discussions around the next iteration of collider experiments are already underway. To provide precision measurements of the known elementary particles and forces, the highest priority option for a future collider is a high-energy lepton collider which can act as a "Higgs factory" [23].

The LEP has been of now largest lepton collider with $\sqrt{s} = 209$ GeV and shut down in 2000 to make way for the LHC built in the same tunnel. To enable efficient precision measurements of the Higgs boson with the next generation of electron-positron colliders, at least $\sqrt{s} = 250$ GeV is needed — the energy for the optimal production cross-section of the Higgs-strahlung process $e^+e^- \to Z^0h$ [24]. At this energy, the Higgs mass m_h can be measured particularly precise as its invariant mass recoils against the reconstructed Z^0 that can be well measured from its lepton decays $Z^0 \to e^+e^-$ and $Z^0 \to \mu^+\mu^-$, regardless of any given h decay mode. However, these Z^0 decays are suppressed in comparison to the hadronic decay ($\mathrm{BR}(Z^0 \to \mu^+\mu^-) \approx \mathrm{BR}(Z^0 \to e^+e^-) \approx 3.4\%$ vs. $\mathrm{BR}(Z^0 \to q\bar{q}) \approx 69.9\%$) [76]. To

utilize the hadronic channel effectively, a very good jet energy resolution of the detector system is necessary — for example using the particle flow approach (see Section 2.3).

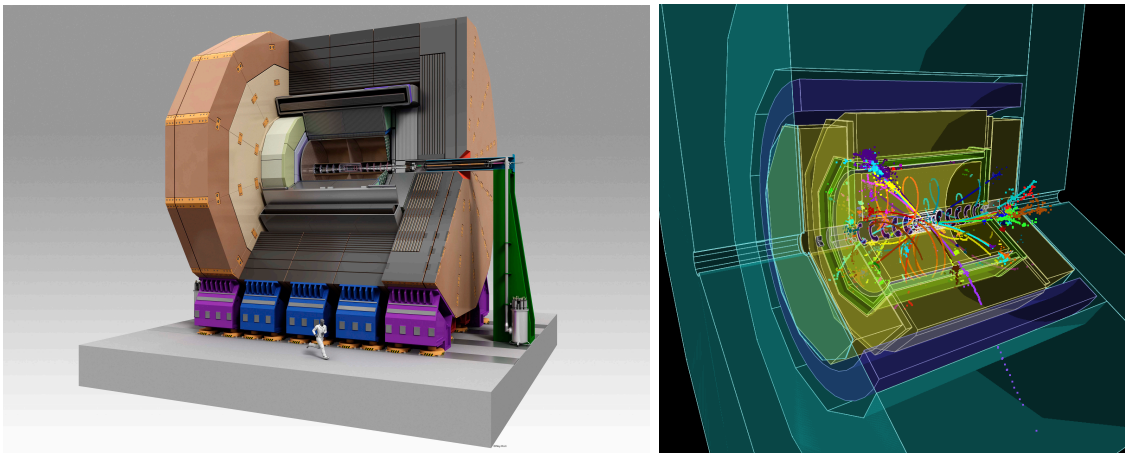
With further increasing energy above $\sqrt{s} \geq 450$ GeV, the WW fusion process $e^+e^- \to \nu\bar{\nu}h$ becomes dominant and allows for measurements of the absolute normalization of the Higgs coupling strengths with a precision at the percent level. At even higher energies, the Higgs-to-top coupling can be studied via the process $e^+e^- \to t\bar{t}h$, which will help to further study how the Higgs mechanism leads to the generation of fermionic masses — as the Higgs-fermion coupling is proportional to the fermion mass (see Equation 1.58). Additionally, the Higgs self-coupling will become available via $e^+e^- \to Zhh$. Further interesting physics processes available at an e^+e^- machine and their cross-sections as functions of $\sqrt{s}$ are shown in Figure 1.3 (note the specified beam polarizations).

Either a very large radius circular collider or a large linear collider would need to be built for this purpose. The advantage of a circular lepton collider would be the reusability of the infrastructure for a future hadron collider, the same way as LEP gave way to the LHC. This would warrant a circular accelerator of 80–100 km. Further, a circular collider allows for a higher instantaneous luminosity, since in a circular storage ring the bunches can be collided over and over again, while in a linear collider, they need to be accelerated anew. It also allows for multiple interaction points and therefore multiple simultaneously running detectors (leading to a multiplication of the achievable integrated luminosity). On the other hand, a linear collider can potentially achieve higher energy with an overall smaller accelerator size since the energy loss through synchrotron radiation can be disregarded. In addition, there are further considerations such as the possibility of setting the beam polarization at linear colliders since the polarization of electrons is preserved during linear acceleration making specific physics processes more dominant. The choice also has an impact on detector design which can be more compact at e^+e^- linear colliders because the application of bunch trains (a collection of bunches) allows to power down parts of the detectors during the downtime between bunch trains ("power-pulsing") and therefore the detector components require less cooling infrastructure.

There are currently multiple proposals for future circular or linear colliders in various stages of planning. These proposals all focus on an electron-positron collider — at least in the initial stage:

- **ILC:** For linear colliders, the most advanced proposal is the *International Linear Collider* (ILC) [24]. The ILC envisions colliding polarized electrons and positrons with an initial baseline of $\sqrt{s} = 250$ GeV, with the possibility of upgrading the accelerator in two stages to $\sqrt{s} = 500$ GeV and $\sqrt{s} = 1$ TeV. The overall length of the collider would be about 20 km for the 250 GeV machine, and 30 to 50 km for the 500 GeV and 1 TeV collider, respectively. The envisioned site for the ILC is in Japan.

  The ILC is designed to host two multi-purpose detectors, the *International Large Detector* (ILC) and the *Silicon Detector* (SiD). Both detectors are designed to achieve a high-resolution jet energy reconstruction and di-jet mass performance. For this purpose event reconstruction is performed using Particle Flow Algorithms (PFA) (see Section 2.3) which require high-granularity calorimeters and very effective tracking



(a) ILD Detector **(b)** Event Display

Figure 1.4: **(a)** Rendering of the envisioned International Large Detector (ILD) detector. **(b)** Event display of the hadronic decay of a simulated $t\bar{t}$ event in the ILD. The color coding corresponds to reconstructed tracks of individual particles. Figures taken from Reference [102].

systems. The concept for the ILD uses a tracking system combined of continuous-readout time-projection chambers and silicon tracking as well as a high-granularity calorimeter contained in a 3.5 T magnetic field. A rendering and an event display of the ILD are shown in Figure 1.4. The SiD detector is focused on a cost-effective design by using mostly silicon both for tracking and in the highly granular calorimeter and will use a 5 T magnet. As there is only one interaction region, only one of the detectors can take data at a time.

- **CLIC:** Another linear collider is proposed to be built at CERN: The *Compact Linear Collider* (CLIC) [103] aims to achieve a multi-TeV collision energy by building the accelerator in three stages. The first stage would aim for $\sqrt{s} = 380$ GeV with a collider length of 11 km, the second stage would achieve $\sqrt{s} = 1.5$ TeV with a length of 29 km, and the third stage $\sqrt{s} = 3$ TeV with a collider length of 50 km. The higher peak energy in comparison to ILC would be achieved by using accelerator gradients of up to 100 MV/m (compared to 35 MV/m at the ILC). There are further differences between the colliders such as the envisioned instantaneous luminosity, bunch spacing and polarization. A single detector, the *CLIC detector* (CLICdet) [104], is planned to study the collisions at the interaction point. The innermost component of CLICdet is a silicon pixel vertex detector surrounded by an all-silicon tracker. The high-granularity ECAL and HCAL are optimized for the particle-flow paradigm. Both are placed inside a 4 T magnet surrounded by the muon system.

- **FCC:** For a circular lepton collider, CERN proposes to build the *Future Circular Collider* (FCC) [105] with a tunnel length of about 100 km. This collider would in its first variant be a e^+e^- collider, dubbed FCC-ee, aiming for up to $\sqrt{s} \approx 350 - 365$ GeV,

i.e. the optimal energy for $t\bar{t}$ pair production. Due to synchrotron radiation, it would be difficult to achieve a much higher energy. At lower energies it would also be used to generate large amounts of Z^0 bosons ($\sqrt{s} \sim 91$ GeV), WW pairs ($\sqrt{s} \sim 160$ GeV) and Higgs boson via the Higgs-strahlung process ($\sqrt{s} \sim 250$ GeV) (see Figure 1.3).

As a baseline, the FCC-ee is planned with two interaction points, each equipped with a general-purpose detector. The current conceptual designs are called *CLIC-Like Detector* (CLD) and *International Detector for Electron-positron Accelerators* (IDEA) [106]. The CLD is similar to CLICdet and utilizes a silicon vertex detector and silicon tracker together with a highly granular 3D-imaging calorimeter (a silicon-tungsten ECAL and a scintillator-steel HCAL). IDEA employs a silicon vertex detector, a short-drift wire chamber as a tracker, and a dual-readout calorimeter (see Section 2.2.2). It also uses SiPMs for read-out and a fine granularity to allow for particle-flow reconstruction. Due to the beam crossing angle of 30 mrad, the detector solenoid magnets of both detectors can have a maximum strength of 2 T as with a higher strength the beam emittance would increase too much leading to a loss in luminosity.

Once the physics potential of the FCC-ee is exploited, the tunnel can be reused to upgrade the machine to a hadron collider, the FCC-hh, aiming for $\sqrt{s} \sim 100$ TeV pp collisions as a discovery vehicle. Just like in the LHC, heavy ion collisions would be possible as well. Further options are an electron-proton collider, dubbed FCC-eh, that could produce 3.5 TeV ep collisions following in the footsteps of HERA.

- **CEPC:** A similar accelerator is proposed in China: the *Circular Electron Positron Collider* (CEPC) [107, 108]. This circular e^+e^- collider is a very similar project to CERN's FCC-ee. It is also envisioned to run it at the four different energy levels up to the $t\bar{t}$ pair production energy at $\sqrt{s} \sim 360$ GeV, employs a circular accelerator with a 100 km circumference, and two interaction points with large general-purpose detectors. For the two detectors, one is envisioned as an adapted ILD design, and the other is a detector based on IDEA. The CEPC also allows for a follow-up proton collider, the *Super Proton Proton Collider* (SPPC), as a high-energy discovery machine.

- **Others:** Other less developed collider concepts include a muon collider [109], the Cool Copper Collider (C^3) [110], or leveraging plasma-wakefield acceleration [111], i.e. in a hybrid version combining plasma-wakefield acceleration for electrons and conventional radio-frequency (RF) acceleration for positrons [112].

All envisioned detectors have in common that they use highly granular calorimeters to enable efficient particle-flow algorithms for event reconstruction. This high granularity also makes simulating the detector response more challenging and requires simulation tools that allow both high fidelity and computational efficiency. To this end, this thesis explores the application of generative models for highly granular calorimeter simulations in Chapter 5 and 6.

1.5 Particle Physics Simulations

The scientific method starts with a theory. Based on the theory a prediction is made on how an experiment might turn out, then the experiment is performed, and the result is compared to the prediction. Based on this result, the theory might be adjusted and the cycle starts anew. This is how scientific progress is made.

Above, we have introduced the particle physics theory with the SM Lagrangian at its heart and discussed how detector experiments at colliders are performed. Considering that the SM Lagrangian is a theory equation that cannot be evaluated analytically and the detector experiment yields essentially sensor data, how can we consolidate the two with a prediction? The answer is a simulated experiment.

The theory — for example, the regular SM Lagrangian or a modified theory — is turned into a prediction by modeling how the detector measurement would look like, if the theory were true. Thereby, this modeled interaction can be compared with the real detector experiment and the theory is confirmed or disproven. For this method to work, the simulations need to match the accuracy of the experiment and with more precise measurements, i.e. due to the application of high-granularity calorimeters, the bar for more accurate simulations rises. At least as many simulated events as real data are required for the results to not be limited by fluctuations due to the size of the simulated samples, hence with higher luminosity colliders, also comes an increasing need for more simulated data which may require more computational resources.

To achieve an accurate modeling of the detector response to a collider interaction, traditionally a pipeline of multiple separate simulation tools is needed to make up the *full simulation*:

- **Event generation:** The event generation is the first step in a simulation chain and models the hard scattering processes between the initial state of the beam particles, i.e. the protons or electrons. These calculations involve multidimensional integrations of the differential cross-section for every process that can occur, which are analytically very difficult. Therefore, event generators apply numerical integration approaches, usually Monte Carlo (MC) methods. The event generator output is a list of particles that are produced in the initial beam particle interaction. Event generation is usually done with tools such as MadGraph [113], WHIZARD [114], PYTHIA [115], and Herwig [116].

- **Hadronization:** Quarks and gluons produced in the initial interaction form hadrons due to color confinement. This step is called *hadronization* and results in a collimated spray of hadrons, known as a jet (see Section 1.6). Their lifetime and decay are also taken into account at this step. Hadronization is implemented in event generators such as PYTHIA.

- **Detector description:** As the produced particles of the interaction as well as their decay products and said jets interact with detector material surrounding the interaction point, a detailed description of the detector is needed. This is done in software such as DD4hep [117], in which the geometry and all material compositions inside the detector are defined.

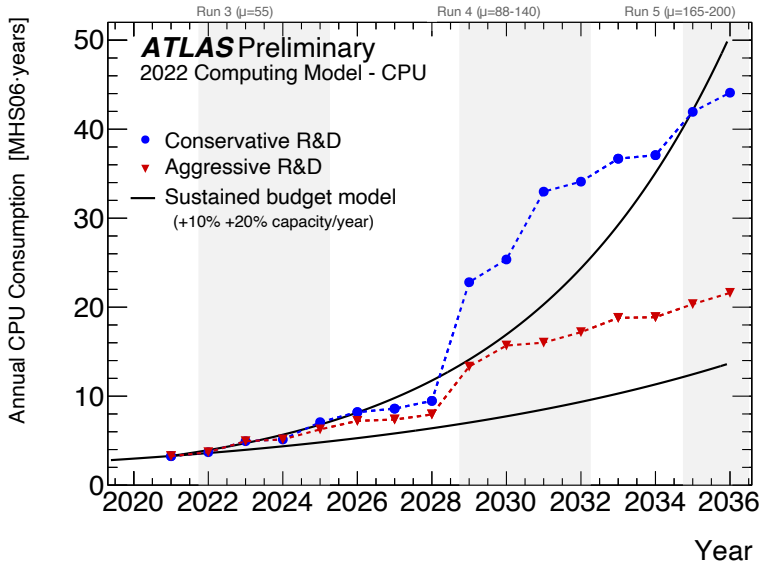


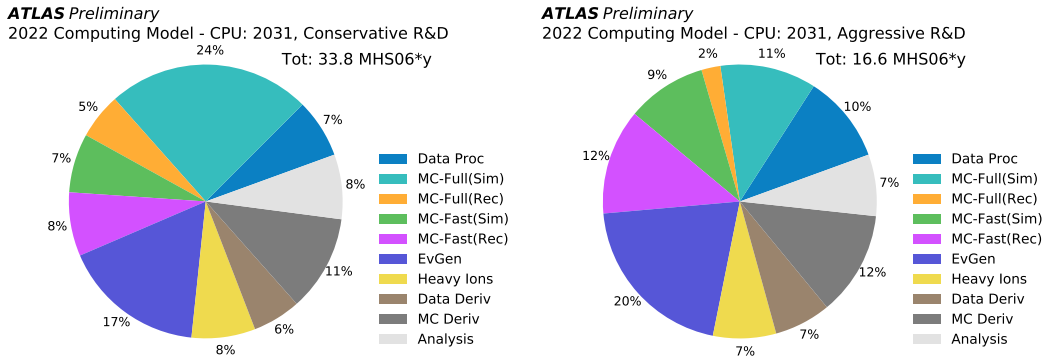
Figure 1.5: Projected evolution of the ATLAS annual CPU consumption from 2020 until 2036 measured in million HS06 [119]. A conservative (blue) and an aggressive (red) R&D scenario are considered and compared to a 10% and a 20% sustained computing capacity increase (including budget increases and improvements of new hardware). Figure taken from Reference [120].

- **Transport software:** Finally, the interaction between particles and matter is simulated using a transport software. At this step, all interactions between particles and the detector matter are simulated, including decays, scattering processes, absorption, and the impact of the magnetic field. The energy depositions in the active material are recorded in the same format as the real experiment. Depending on how many particles enter the detector and the detector volume, this step of the simulation chain can be very computationally intensive. The most common transport software in high-energy physics is GEANT4 [118]. A more detailed introduction to detector simulation is given in Section 2.4 with the example of calorimeter simulations.

Now the simulated events can be analyzed with the same software infrastructure as the real data.

In comparison to a full simulation, a *fast simulation* uses tools to skip or alter parts of the full simulation for increased computational efficiency albeit at an often lesser accuracy. Various fast simulation methods are discussed in Section 2.4. Among the methods for fast simulations are generative machine learning models, the topic of this thesis.

With the HL-LHC upgrade and its increased luminosity, the computational requirements for simulations will likely increase significantly to still achieve equal simulated and real data statistics. An extrapolation of the annual CPU consumption required by the ATLAS experiment is shown in Figure 1.5. A stark increase is projected after the HL-LHC starts operation in 2028. This computing usage includes all computational needs of the experiment, including event reconstruction, event generation, analysis, data processing, and MC simulation.



(a) Conservative R&D **(b)** Aggressive R&D

Figure 1.6: Breakdown of the projected CPU computing usage by ATLAS during the LHC's fourth period of operation (Run 4) in 2031 for either a conservative **(a)** or an aggressive **(b)** R&D scenario. Note that event generation (EvGen) is here separated from the detector simulation (MC). Figures taken from Reference [120].

Aggressive R&D efforts are needed to stay within a sustained budget.

A breakdown of the projected CPU computing usage by ATLAS in 2031 is shown in Figure 1.6. A large portion of the computing usage is driven by the MC simulation. With a conservative R&D scenario, the total computing usage of MC simulations (both full simulation and fast simulation, without event generation) would take up 31% of the total budget and with aggressive R&D it would use up 20% (of an overall about 50% lower budget). One should note, that even the conservative R&D scenario includes already substantial improvements in the fast calorimeter and tracker simulations [120], i.e. without using novel fast simulation methods, the computing usage of simulations would be even higher. To achieve such improvements, this thesis explores the use of generative machine learning models for fast particle physics simulations.

1.6 Jet Physics

Collisions at hadron colliders produce almost always quarks and gluons and also on lepton colliders the final state often contains quarks. The strong interaction between them causes the creation of further quarks and anti-quarks pairs. This process is known as hadronization and happens on a distance scale of 10^{-15} m. Hadronization of quarks and gluons take the form of a parton shower and the collimated stream of hadrons can be measured as a *jet*, since due to color confinement the quarks are never observed freely. Each quark produced in the initial collision can produce a jet, so an event like $e^+e^- \to q\bar{q}$ would produce two jets directed in opposite directions in the center-of-mass frame. A detailed introduction to jet physics at the LHC can be found in Reference [121].

A jet contains on average 60% of its energy in the form of charged particles (like $\pi^{\pm}$), 30% as photons (from $\pi^0 \to \gamma\gamma$ decays), and 10% as neutral particles like neutrons. In traditional calorimeters, individual particles in the jet cannot be resolved and the jet energy

is determined by summing up the energy deposited in the ECAL and HCAL. This may change with future collider experiments featuring high-granularity calorimeters optimized for particle-flow reconstruction (see Section 2.3).

Qualitatively the hadronization process can be split into five steps: [68]

1. The produced $q\bar{q}$ separate at a very high velocity.
2. A color field emerges between them with an energy density of ~ 1 GeV/fm.
3. With increasing distance between the original quarks the energy of the color field rises until enough energy is accumulated to form new $q\bar{q}$ pairs.
4. Between the new $q\bar{q}$ pairs further color fields emerge. This creates a cascade of secondary pairs.
5. The cascade ends once the quarks and antiquarks have lost enough energy to form hadrons. Unstable hadrons decay further until only stable particles remain.

How exactly this process works is still unknown. However, there are many models that aim to simulate the hadronization process. These are implemented in simulation tools such as PYTHIA and HERWIG.

At collider experiments, a precise measurement of the jets allows the study of the initiating particles. Jet clustering algorithms are used to reconstruct the four-momentum of the jet. In addition to the four-momentum, several observables can be derived from the jet substructure that facilitate distinguishing whether they originate from light quarks, gluon, or contain decay products of heavier particles. A detailed introduction to jet physics at the LHC can be found in Reference [121].

1.6.1 Jet Simulation

Since the simulation of jets is a highly complex process, an event generator such as PYTHIA factorizes the generation process into several components. The most important components for the simulation of a $pp \to t\bar{t}$ event are illustrated in Figure 1.7. Typically, the processes are categorized according to the "hardness" scale measured in terms of the transverse momentum p_{T}. The hardness decreases with the radius around the hardest scattering ($\mathrm{d}\hat{\sigma}_0$). Several sub-processes are outlined below in the order of decreasing hardness following Reference [122].

The hardest process is the scattering of the two partons into several outgoing particles. The initial partons are chosen according to the parton distribution functions (PDFs) of each incoming proton and the outgoing particle kinematics are determined by the matrix elements calculated via perturbation theory. It is common to use an external generator like MADGRAPH for the parton-level computations. If short-lived resonances such as the top quarks are produced, they decay almost immediately into other particles.

Several radiative processes can occur in the next step. Fixed-order radiative corrections are considered via matrix-element corrections, matching, and merging strategies. The range where these corrections are valid is shaded violet around the initial hard process in Figure 1.7. Initial-state radiation (ISR) of additional particles originates from soft gluon emissions and

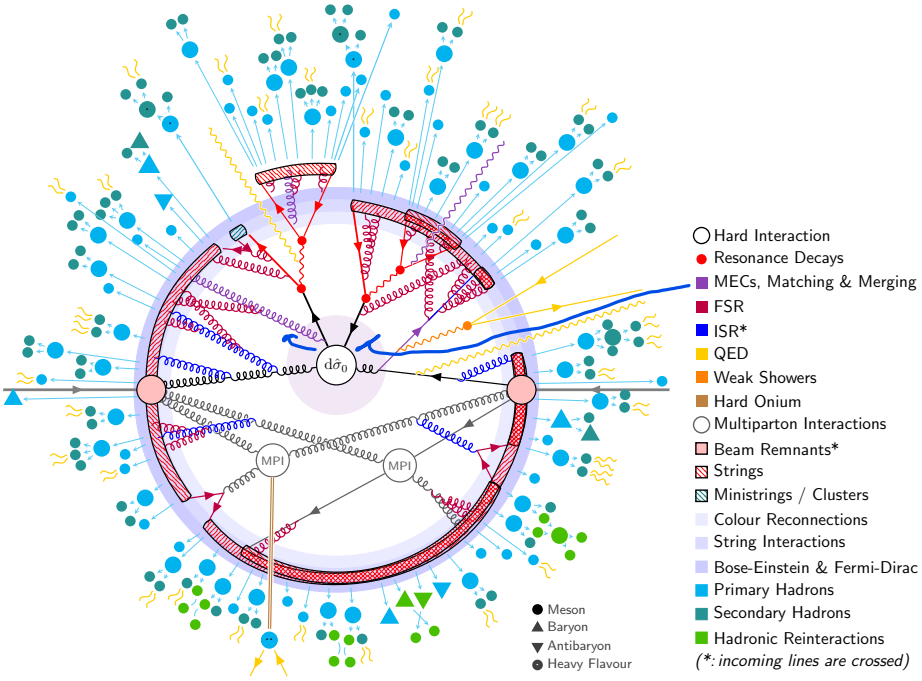


Figure 1.7: Illustration of a $pp \to t\bar{t}$ event as modeled by PYTHIA. Figure taken from Reference [122].

final-state radiation (FSR) produces additional particles from the initial hard scattering or the subsequent resonance decays. In the same range, multi parton interactions (MPI) through scattering with the additional partons of the incoming protons can occur.

A bit farther away, around the blue shaded ring of Figure 1.7, the hadronization process discussed above begins with the formation of color strings between the outgoing partons. These strings are color singlets that confine the QCD partons due to the strong interaction. This is where the transition from the partonic to the hadronic phase is modeled. In PYTHIA, the Lund string model [123] is used to describe the fragmentation of the strings into hadrons. If an unstable hadron is produced, it decays further until only stable particles remain. These particles however may further interact by scattering or annihilation processes. The output of the event generator is given as a list of particles with their properties such as four-momenta and particle types. This list can then be used as input for the detector simulation or directly processed further for example by a jet clustering algorithm. Details about all of these processes can be found in References [122].

1.6.2 Jet Algorithms

Jet algorithms are used to cluster measured objects (here termed 'particles' for the sake of simplicity) together to reconstruct a jet. There are two types of jet algorithms: cone algorithms and sequential-recombination algorithms. *Cone algorithms* cluster particles within a specific conical angular region, while *sequential-recombination algorithms* iteratively combine particles that are closest together in some distance measure. At the LHC and likely also at

future lepton colliders, the latter algorithms are primarily used.

Infared and collinear (IRC) safety is an important constraint for jet algorithms. IRC safety implies, that the clustered jet is unchanged after the emission of infinitely soft particles and under the collinear splitting or merging of particles within the jet. Further constraints on the algorithms include invariance to longitudinal boosts (ensured by using the rapidity y instead of the pseudorapidity η in the computations), independence of detector details, and insensitivity to pile-up.

The distance measure for the sequential-recombination algorithms used at the LHC is defined between a particle i and j as

$$d_{i,j} = \min\left(p_{\mathrm{T},i}^{2p}, p_{\mathrm{T},j}^{2p}\right) \frac{\Delta R_{ij}^2}{R^2} \tag{1.62}$$

where $p_{\mathrm{T},i}$ and $p_{\mathrm{T},j}$ are the transverse momenta of the particles, $\Delta R_{ij} = \sqrt{(y_i - y_j)^2 + (\phi_i - \phi_j)^2}$ is the angular distance in the y-ϕ plane, and R is a parameter that defines the jet radius. A second distance measure is defined between the particle i and the beam axis as

$$d_{i,\mathrm{B}} = p_{\mathrm{T},i}^{2p}. \tag{1.63}$$

The parameter p determines the order in which low- and high-p_{T} particles are clustered: $p = 1$ for the k_t *algorithm* [124], $p = 0$ for the *Cambridge-Aachen algorithm* [125], and $p = -1$ for the *anti-k_t algorithm* [126] (the most commonly used one).

The clustering is performed by finding the minimal distance $d_{\min}$ between all distances $d_{i,j}$ and $d_{i,\mathrm{B}}$. If $d_{\min} \in \{d_{i,j}\}$, particles i and j are combined by four-momentum addition into a new pseudo-particle and particles i and j are removed. If $d_{\min} \in \{d_{i,\mathrm{B}}\}$, particle i is declared a jet and removed. This process is repeated until all particles are clustered into jets.

At a future high-energy lepton collider, the jet clustering algorithms will need to be optimized to take full advantage of the collider and detectors. [127] Challenges for jet reconstruction include the amount of jet final states, pile-up background, and initial state radiation. It has been shown that the *Valencia algorithm* (VLC) [128] outperforms traditional e^+e^- jet algorithms and k_t-like algorithms at high energy lepton colliders. The VLC algorithm is also a sequential-recombination algorithm, but it uses a different distance measure compared to the k_t algorithm family. The VLC distance measures are defined as:

$$d_{i,j} = 2\min\left(E_i^{2\beta}, E_j^{2\beta}\right) \frac{1 - \cos\theta_{ij}}{R^2} \tag{1.64}$$

and

$$d_{i,\mathrm{B}} = E_i^{2\beta} \sin^{2\gamma}\theta_{i\mathrm{B}} \tag{1.65}$$

where E_i and E_j are the energies of the particles, θ_{ij} is the angle between the particles, and $\theta_{i\mathrm{B}}$ is the angle between the particle and the beam axis. The parameter β is set to $\beta = 1$ to value the softer of the two particles higher (like the k_t algorithm). When setting also the parameter $\gamma = 1$, the beam distance becomes $d_{i,\mathrm{B}} = E_i^2 \sin^2\theta_{i\mathrm{B}} = p_{\mathrm{T},i}^2$, which is the same as in k_t algorithm. Compared to the k_t algorithm, VLC replaces essentially the transverse momenta with the particle energies and the angular distance with $1 - \cos\phi_{ij}$ — choices that seem sensible for a lepton collider where the collision energy is clearly defined.

1.6.3 Jet Substructure

Not only the study of whole jets but also their internal structure has become more important over the recent years. This is motivated by the LHC collision energies being high enough to frequently produce electroweak resonances, such as top quarks, $W^{\pm}/Z^0$ bosons, and Higgs bosons, with transverse momenta way beyond their rest mass. Since these resonances decay predominantly into quarks, they produce boosted jets which are highly collimated and overlapping in the lab frame. This leads to a very large jet, a so-called *fat jet*.

To distinguish whether these jets originate from interesting electroweak resonances or the large QCD background, i.e. jets produced from high-energy quarks and gluons (excluding top quarks), physicists have started to study the internal structure of jets. These studies are enabled by the popularization of the above-discussed sequential-recombination algorithms since they retain the whole history of jet clustering. This study of the internal structure of jets is known as *jet substructure* analysis and several observables have been explored to help in its understanding. Some of these are used for the studies presented in Chapter 7 and introduced in the following. A good general introduction to jet substructure and boosted-object phenomenology can be found in Reference [129].

N-subjettiness

N-subjettiness [130] encompasses a family of jet shape variables denoted τ_N, where N is the number of subjets the jet consists of. It is defined as:

$$\tau_N = \frac{1}{d_0} \sum_k p_{\mathrm{T},k} \min \left(\Delta R_{1,k}, \Delta R_{2,k}, \ldots, \Delta R_{N,k} \right) \tag{1.66}$$

where k runs over all the particles in the reconstructed jets, $p_{\mathrm{T},k}$ are the transverse momenta of the particles, and $\Delta R_{J,k}$ is the angular distance between the jet axis of candidate subjet J and the particle k. The angular distance in the rapidity-azimuth plane is given by $\Delta R_{J,k} = \sqrt{(y_J - y_k)^2 + (\phi_J - \phi_k)^2}$. Here, d_0 is a normalization factor given by

$$d_0 = \sum_k p_{\mathrm{T},k} R_0 \tag{1.67}$$

where R_0 is the jet radius parameter of the original jet clustering algorithm. By minimizing τ_N the subjet axes are found.

Jets with $\tau_N \approx 0$ have all particles aligned with the N (or fewer) subjets. Jets with $\tau_N >> 0$ contain a lot of particles farther away from the jet axis and have at least $N+1$ subjets. Therefore, one can interpret τ_N as a measure of the radiation distribution around the subjet axes. Since the value of τ_N is larger for jets originating by gluons, the N-subjettiness ratio

$$\tau_{N,N-1} = \frac{\tau_N}{\tau_{N-1}} \tag{1.68}$$

is a good observable to distinguish N-pronged jets from QCD background jets. In particular, $\tau_{2,1}$ can be used to distinguish $W/Z/H$ jets from QCD jets and $\tau_{3,2}$ can be used to discriminate top jets.

Energy Flow Polynomials

Energy Flow Polynomials (EFPs) [131] are a set of jet substructure observables that form a linear basis for all IRC-safe observables. They can be expressed in the language of graph theory and for a multigraph G with N vertices and edges $(k, l) \in G$ an EFP is defined as

$$\mathrm{EFP}_G = \sum_{i_1=1}^{M} \cdots \sum_{i_N=1}^{M} z_{i_1} \cdots z_{i_N} \prod_{(k,l)\in G} \theta_{i_k i_l} \tag{1.69}$$

where the jet is made up of M particles, $z_i = E_i / \sum_{j=1}^{M} E_j$ is the energy fraction of particle i, and θ_{ij} is the angular distance between particles i and j. At hadron colliders, z_i is the fraction of transverse momentum and the angular distance is expressed in the rapidity-azimuth plane.

It can be shown that many common jet observables can be expressed as a linear combination of EFPs. Depending on the application, a certain set of EFPs can be used to distinguish various jet topologies. In principle, there are an unlimited number of EFPs (for the limit of jets with an infinite number of particles), but in practice, only the ones with few vertices and edges are efficient to compute.

Energy Correlator Functions

Energy Correlator Functions (ECFs) [132] are another set of jet substructure variables based on the energies of particles and the angles between them. Although introduced earlier, they can be viewed as a special case of EFPs with complete graphs — a graph where every vertex is connected by a unique edge to every other vertex. The ECFs achieve similar discriminating power as N-subjettiness but do not require the clustering history.

The 2-point correlators are well suited for quark/gluon discrimination and the 3-point correlators for boosted $W/Z/H$ identification. They are defined as

$$e_2^{(\beta)} = \sum_{i<j\in J} z_i z_j \Delta R_{ij}^{\beta} \tag{1.70}$$

and

$$e_3^{(\beta)} = \sum_{i<j<k\in J} z_i z_j z_k \Delta R_{ij}^{\beta} \Delta R_{ik}^{\beta} \Delta R_{jk}^{\beta} \tag{1.71}$$

where $z_i = p_{\mathrm{T},i} / \sum_j p_{\mathrm{T},j}$ is the transverse momentum fraction of particle i and ΔR_{ij} is the angular distance between particles i and j. i, j, and k are particles in jet J. The exponent is usually set to $\beta = 1$. Similar to the N-subjettiness, ratios of ECFs are particularly sensitive to discriminate boosted massive particles from QCD jets. A common ratio with large discriminating power is

$$D_2^{(\beta)} = \frac{e_3^{(\beta)}}{(e_2^{(\beta)})^3}. \tag{1.72}$$

Chapter 2

Calorimetry

In particle physics, calorimetry is the destructive measurement of a particle's energy by its absorption in matter. When a particle interacts with matter, it can deposit energy via electromagnetic or hard-/hadronic interactions in the material. With a suitable material, this interaction can be measured and the deposited energy calculated. In modern collider experiments, a calorimeter has several important tasks, including event selection, triggering, precision measurements of individual particles and jets, and measuring the energy flow in the events, for example, to determine missing energy.

This chapter introduces the basic concepts of modern calorimetry as well as novel algorithms and simulation methods that are used to advance calorimeters for future experiments. Although most of the results in the following chapters were derived from electromagnetic showers, an introduction to hadronic showers and hadronic calorimeters is given as well. In particular, hadronic showers are the main motivation for particle flow algorithms, which in turn are the incentive for the application of high-granularity calorimeters, which are simulated in the Chapters 5 and 6.

In Section 2.1, the basic electromagnetic and hadronic interactions of particles with matter are discussed. Section 2.2 discusses the types of calorimeters and their response to different particles. The particle flow approach to calorimetry is explained in Section 2.3. Finally, full physics-based as well as fast simulations of calorimeters are introduced in Section 2.4.

A good overview of calorimetry can be found in Reference [133] with a more comprehensive introduction found in Reference [134]. A very detailed book on calorimetry is Reference [135].

2.1 Particle-Matter Interaction

During the interaction of particles with matter, particles can create cascades of additional particles. These cascades of secondary particles are called *particle showers*. Usually one differentiates between *electromagnetic showers* and *hadronic showers*, depending on the type of fundamental interaction occurring.

Electromagnetic showers are initiated by electrons, positrons, and photons, while hadronic showers result from interactions with hadrons, such as protons and pions. The possible interactions and showering behavior are rather distinct, therefore they are discussed separately.

2.1.1 Electromagnetic Interactions

Several well-understood processes play a role in the formation of electromagnetic showers. Electrons and positrons lose energy mainly via ionization and radiation (bremsstrahlung). Ionization occurs when a particle's energy is sufficient to release the atomic electrons from the Coulomb fields which are generated by the nuclei of the medium. Many particle detectors function with ionization as these free electrons can yield an electric signal. Another way to detect the energy deposition is using scintillation light. This occurs when charged particles excite atoms in the scintillator material without ionizing them. The de-excitation results in released photons that can be picked up as a light source from which a calorimeter signal is recorded. Cherenkov light is also sometimes used for calorimeters. This radiation occurs when charged particles travel in a medium faster than that medium's speed of light. At high energies, bremsstrahlung is a dominant electromagnetic interaction and to a lesser extent energetic knock-on electrons, so-called δ-rays. At very high energies, even nuclear reactions through electromagnetic interactions are possible.

Electrons and Positrons

At energies above 10–100 MeV (material-dependent), bremsstrahlung is by far the dominant form of energy loss by electrons and positrons. It is induced by the Coulomb interaction with the electric fields of nuclei when the e^+/e^- traverse matter. In general, any charged particle traversing matter may undergo bremsstrahlung, though for heavier particles it becomes dominant only at much higher energies. At lower energies, ionization becomes the primary source of energy loss for e^+/e^-, with Møller scattering (for e^-), Bhabha scattering (for e^+) and e^+e^- annihilation contributing to a lesser extent at low energies. These different contributions to the energy loss of electrons and positrons are shown in Figure 2.1a as a function of energy.

The energy at which bremsstrahlung and ionization play an equal role is the material-dependent *critical energy*. It depends on the electron density in the medium which in turn is roughly proportional to the material's atomic number Z. The critical energy[1] can be expressed as [76]

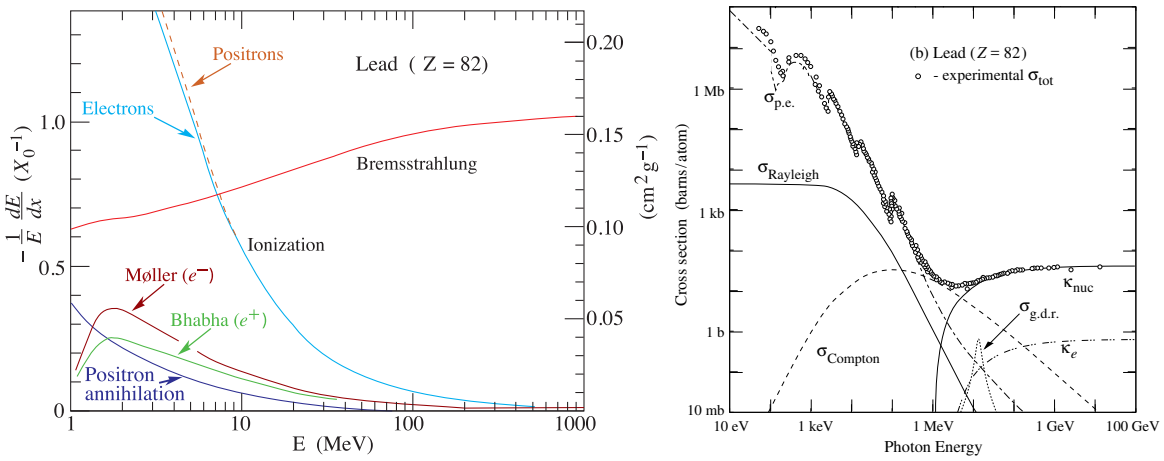
$$\varepsilon_c = \frac{610 \text{ MeV}}{Z + 1.24} \quad \text{and} \quad \varepsilon_c = \frac{710 \text{ MeV}}{Z + 0.92} \tag{2.1}$$

in solid/liquid materials and gaseous materials, respectively. For iron, the critical energy is approximately 21 MeV.

Photons

Photons interact mainly by four different processes: the photoelectric effect, coherent (Rayleigh) scattering, incoherent (Compton) scattering, and electron-positron pair production. Photo-nuclear reactions play a minor role in only a small range of photon energies. The

[1] This parameterization is based on an alternative definition of the critical energy from Reference [136] that defines it as the energy at which the ionization loss per radiation length is equal to the electron energy. It has been found to describe the transverse electromagnetic shower development more accurately [76].



(a) e^+/e^- energy loss **(b)** γ cross-sections

Figure 2.1: **(a)** Processes for energy loss of electrons and positrons in lead as a function of energy. **(b)** Photon cross-sections in lead as a function of energy. $\sigma_{\text{p.e.}}$ is the photo-electric cross-section. $\sigma_{\text{g.d.r}}$ is the photo-nuclear cross-section ("giant dipole resonance"). κ_{nuc} is the cross-section of pair production in a nuclear field and κ_{e} is the pair production cross-section in an electron field. Figures taken from Reference [76].

cross-sections for these interactions are material-dependent. For lead, they are shown in Figure 2.1b.

The *photo-electric effect* is dominant at low energies and its cross-section decreases with the photon energy as E^{-3}. In this process, an atom is excited by the photon and emits an electron. The atom gets de-excited by emitting X-rays or Auger electrons. Its cross-section is dependent on the number of available electrons of the atom and is therefore highly sensitive to the Z value of the medium.

Rayleigh scattering is a coherent process that is also important at low energies. Here a photon is deflected by atomic electrons without any energy loss. Hence, Rayleigh scattering does not result in any energy deposition in the medium, but it can affect the spatial distribution of energy deposits through subsequent processes.

Compton scattering on the other hand describes the scattering of a photon by an atomic electron with energy and momentum transfer. The momentum transfer is large enough to release the electron into an unbound state. In most materials, Compton scattering is dominant for photons in the energy range between a few hundred keV and about 5 MeV. The process also causes the spatial distribution of energy in the medium as the recoiled electrons are scattered in the forward hemisphere of the original photon direction. Often, multiple sequential Compton scattering events lead to a reduction of an initial photon energy in the MeV range to the keV range, resulting in the absorption the photon via the photo-electric effect.

At photon energies of at least twice the electron rest mass, i.e. $E_\gamma \gtrsim 1$ MeV, a photon traversing the field of a charged particle can result in the *pair production* of an electron-positron pair. Most pair productions are caused by the electromagnetic field of a nucleus;

the electron field plays a role only at very high photon energies. The generated electron and positron can in turn initiate processes like bremsstrahlung and ionization. Eventually, the positron will (most likely) annihilate with an electron resulting in new photons and the electron may be absorbed by an ion. As the cross-section of the photo-electric effect and Compton scattering both decrease with the energy, at very high energies, pair-production is the dominant process.

Further, *photo-nuclear reactions* can play a small role at energies around 5–20 MeV. These are reactions such as γp, γn, or photo-induced nuclear fission. The maximum of the photo-nuclear cross-section is called the "giant dipole resonance". At this resonance, the photon energy is equal to the marginal binding energy of the proton or neutron, i.e. the binding energy difference the nucleus would have with one nucleon less.

Both photons and charged particles like electrons traverse matter and are eventually absorbed. There are similarities in how both types of particles interact, i.e. for both the interaction with atomic electrons is important. Yet the cross-sections for the interactions are very different: For charged particles, the cross-sections are in the range of 10^7 to 10^8 barns. For photons, they can be 3 to 5 orders of magnitude smaller. This results in electrons and positrons being much more likely to interact with matter than photons are. Electrons traversing matter lose continuously energy via ionization and bremsstrahlung, i.e. a multi-GeV electron radiates thousands of photons by traversing just 1 cm of lead with most of these photons being very soft in the eV to MeV range. In total, it would lose about 83% of its energy. On the other hand, a multi-GeV photon traversing a centimeter of lead might not interact at all. Based on the *mean free path* λ_γ of the photon, one can calculate that there is about 75% chance for the photon to undergo pair production in this material. [134]

Charged Heavy Particles

The critical energy scales with the particle mass as $(m/m_e)^2$. Therefore, for any particle other than e, the critical energy is significantly higher and bremsstrahlung becomes only a consideration at very high energies. For muons ($m_\mu \approx 207 m_e$), the critical energy is $(m_\mu/m_e)^2 \approx 40,000$ times larger in the same material. Hence, at the same energy, electrons and muons behave very differently. At energies below 100 GeV, the primary energy loss of muons in all absorber materials is due to ionization and δ-rays. This results in an energy loss of about 1–2 $\mathrm{MeV\,g^{-1}cm^{-1}}$, i.e. muons traverse a lot of material without losing much energy. For ionization by charged heavy particles such as muons, the mean energy loss per unit path length $\langle dE/dx \rangle$ (also known as the *mass stopping power*, *specific ionization*, or *ionization density*) is given by the *Bethe-Bloch equation* [76]:

$$-\left\langle \frac{dE}{dx} \right\rangle = K z^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \ln \frac{2 m_e c^2 \beta^2 \gamma^2 T_{\max}}{I^2} - \beta^2 - \frac{\delta}{2} \right] \tag{2.2}$$

where $K = 4\pi N_A r_e^2 m_e c^2$ is a proportionality constant, β is the velocity, γ is the Lorentz factor, $T_{\max}$ indicates the maximum kinetic energy transferred to an electron per collision, I represents the mean excitation energy of the traversed material, and $\delta/2$ describes a correction

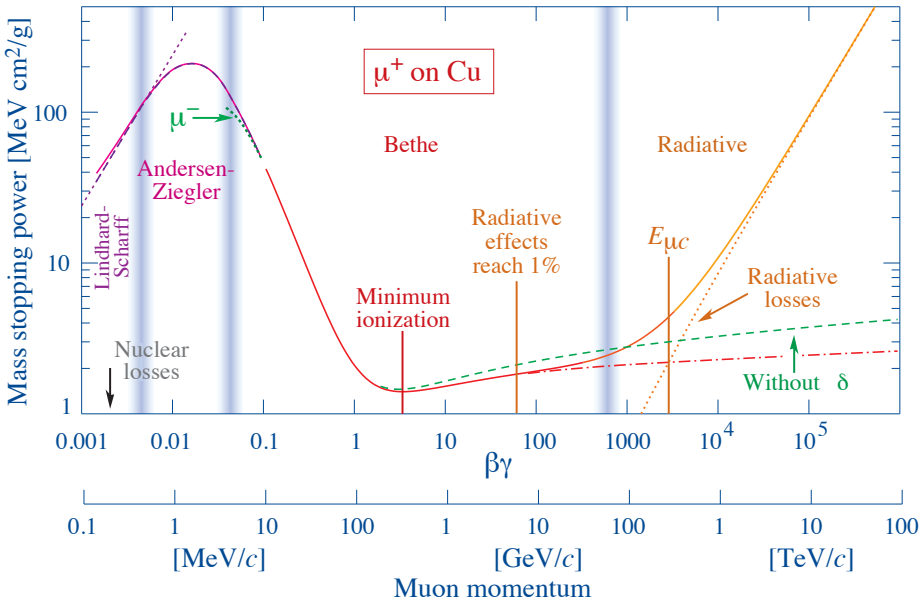


Figure 2.2: Mean energy loss per unit path length $\langle dE/dx \rangle$ for positive muons in copper. The vertical sections indicate different models. Between $0.1 < \beta\gamma < 1000$ the behavior is described by Equation 2.2. Figure taken from Reference [76].

term for the *density effect* — the effect that limits the electric field extension, that occurs with large particle energies, due to material polarization. This equation is accurate to a few percent in the region $0.1 < \beta\gamma < 1000$. In Figure 2.2 the behavior of $\langle dE/dx \rangle$ is shown for positive muons in copper as a function of $\beta\gamma = \frac{p}{mc}$. Charged particles, such as muons and pions, with an energy equal to the curves' minimum around $\beta\gamma \sim 2 - 4$ are called *minimum ionizing particles (MIPs)* as they lose the minimum amount of energy via the ionization interaction.

Considering relatively thin layers of active material present in modern sampling calorimeters, the total energy loss $\Delta E/\Delta x$ for a charged heavy particle usually is quite different from what might be calculated from $\langle dE/dx \rangle$. The measured energy loss distribution in a thin calorimeter has its maximum (its most probable value (MPV)) about 60 % below its mean and a long tail towards larger energy losses, i.e. it follows a Landau-Vavilov distribution. This is mainly due to MIPs losing a comparatively small amount of energy, but this interaction can occur multiple times resulting in many energy depositions around the minimum ionization energy. The high energy tail is due to the small number of collisions with atomic electrons (δ-electrons) resulting in very large fluctuations in the energy transferred in such collisions, bremsstrahlung photons that can induce small electromagnetic showers and nuclear interactions with localized high energy deposits. As the MPV of the energy loss distribution is less dependent on the particle's momentum than the mean of the distribution it is used as a natural scale of the energy deposition for calibration. Such calibration is usually performed with muons as the strength of their interactions is small enough to traverse the material while losing minimal energy through ionization. However, every charged particle traversing the medium around its minimum ionization momentum can result in an excess of MIP energy

depositions (as can be observed for example in Figure 5.7).

2.1.2 Electromagnetic Showers

These fundamental electromagnetic interactions happen many times while charged particles and photons traverse the medium. A primary multi-GeV electron might enter the calorimeter and radiate a large number of photons on its way through via bremsstrahlung. Most of these photons are very soft, i.e. carry little energy, and are hence absorbed via Compton scattering or the photo-electric effect, while slightly harder multi-MeV photons generate e^+e^- pairs. The generated electrons and positrons in turn radiate more photons and this cycle continues. This results in what is known as an *electromagnetic shower*: a cascade of various electrons, positrons, and photons. The number of particles generated in a shower induced by an electron of energy E_0 can be estimated with $N \approx \frac{E_0}{\varepsilon_c}$.

The shower develops primarily in the direction of the original particle, with the energy deposited first increasing towards a maximum, the *shower maximum*, and then decreasing again until every particle is absorbed in the material — or leaves the calorimeter material (*shower leakage*). The initial increase is due to the particle multiplying once the cascade starts. However, this leads to each particle carrying on average less energy as the shower progresses and the likelihood of pair production by photons decreases as does the likelihood of bremsstrahlung by electrons/positrons. Thus, the particle multiplication slows down and beyond the shower maximum, the number of particles and the energy deposited decrease. The mean shower maximum x can be parameterized as [135]

$$\frac{x}{X_0} = \ln\left(\frac{E_0}{\varepsilon_c}\right) + C \tag{2.3}$$

where X_0 is the radiation length of the material (see below) and C is a constant equalling $C = -0.5$ for an electron as a primary particle and $C = 0.5$ for photons.

Two common material-independent variables for describing how electromagnetic showers develop in a medium are the *radiation length* X_0 for the longitudinal shower development (in the incident particle direction) and the *Molière radius* ρ_{M} for the transverse shower development. Note that, using these variables, the shower development can be expressed in a material-independent way, while the values of the variables are material-dependent.

The *radiation length* X_0 is defined as the longitudinal distance in which high-energy electrons/positrons lose $(1 - e^{-1}) = 63.2\%$ of their initial energy via bremsstrahlung. The radiation length for a given medium is material-dependent and can be approximated by [76]

$$X_0 = \frac{716.4\ A}{Z(Z+1)\ln(287/\sqrt{Z})}\,\mathrm{g\,cm^{-2}}. \tag{2.4}$$

The asymptotic cross-section for photon interactions can be related to the radiation length by [134]

$$\sigma(E \to \infty) = \frac{7}{9}\frac{A}{N_{\mathrm{A}} X_0} \tag{2.5}$$

where the ratio of Avogadro's number N_{A} and the atomic mass A is the number of atoms per gram. Hence, *mean free path* λ_γ of very-high-energy photons, i.e. the average longitudinal

distance traveled by a photon before interacting, can be expressed as

$$\lambda_\gamma = \frac{9}{7} X_0. \tag{2.6}$$

The *Molière radius* ρ_M describes the transverse shower development and is defined as [134]

$$\rho_\mathrm{M} = E_\mathrm{s} \frac{X_0}{\varepsilon_c} \tag{2.7}$$

where the scale energy E_s is given by $E_\mathrm{s} = m_e c^s \sqrt{4\pi/\alpha} = 21.2$ MeV. Typically, 85–90% of the shower energy is contained within a cylinder of radius ρ_M around the longitudinal shower axis.

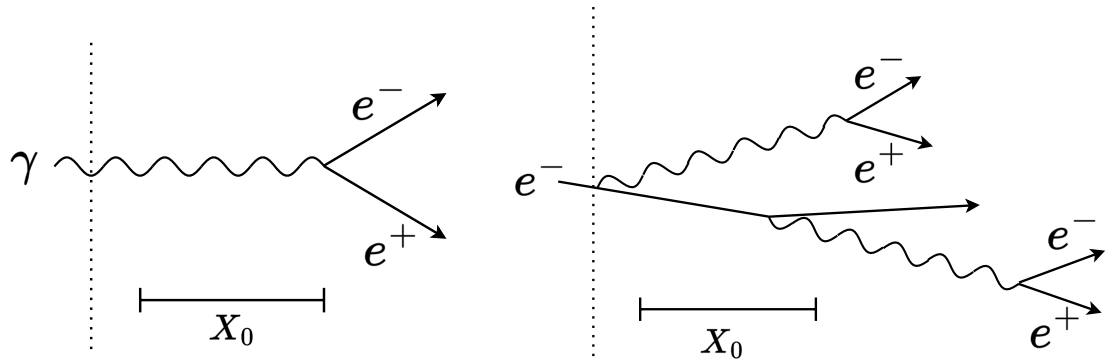
Interestingly, the Molière radius is less material-dependent than the radiation length. This can be seen when comparing Equation 2.4 with Equation 2.7 and 2.1. The impact of the material variables A and Z is less pronounced in the Molière radius. As an example, one can compare copper (Z=29) and lead (Z=82). The radiation length in copper is $X_0 = 14.3$ mm and in lead $X_0 = 5.6$ mm, i.e. a factor of 2.5 difference. So to contain the same shower, 2.5 times thicker copper is needed compared to using lead. On the other hand, the Molière radius in copper is $\rho_\mathrm{M} = 15.2$ mm and in lead $\rho_\mathrm{M} = 16.0$ mm, i.e. a factor of 1.05 difference implying similar shower containment in the lateral direction [134]. Both the radiation length and the Molière radius can be approximated also for mixtures and compounds. Details can be found in Reference [135].

As discussed above, the cross-sections for electrons/positrons and photons are quite different. This also leads to a difference in the shower development depending on the original primary particle type. Once an electron enters the calorimeter, it looses continuously energy through bremsstrahlung and within one radiation length, it has lost on average 63.2% of its energy. High-energy photons, however, travel on average 9/7 radiation lengths until even the first interaction occurs. Therefore, comparing photon and electron showers induced by a particle with the same energy, the photon shower will (on average) develop with a shower maximum deeper in the material. Further, the shower depth fluctuates more with photon showers as electrons interact immediately, bremsstrahlung occurs continuously, and the first photon interaction varies quite a bit. A shower starting through an incident high-energy photon or electron is illustrated in Figure 2.3.

2.1.3 Hadronic Interactions

In the absorption of high-energy hadrons, the strong interaction plays an important role, together with electromagnetism, the weak force, and gravity. Several processes can occur when the hadron traverses the medium. Similar to muons, charged hadrons ionize atoms right from the start when entering the material. However, unlike muons, the hadron will eventually interact strongly with an atomic nucleus, which may result in a variety of processes. For neutral hadrons, such as neutrons generated in the course of hadronic showers, these nuclear interactions are the only way for them to lose energy as they do not ionize the medium.

The average distance the hadron travels until a nuclear interaction occurs, i.e. the mean free path, is called the *nuclear interaction length* λ_int. It is material-dependent and is



(a) Incident photon **(b)** Incident electron

Figure 2.3: Start of an electromagnetic shower induced by a high-energy photon **(a)** or by a high-energy electron **(b)** once they enter an absorber material.

approximately given by [134]

$$\lambda_{\text{int}} = \frac{A}{N_{\text{A}}\sigma_{\text{tot}}} \tag{2.8}$$

where σ_{tot} is the total cross-section for nuclear interactions. Usually, the proton cross-section is used, although the interaction length also depends on the composition of the incident hadron, i.e. mesons have a smaller λ_{int} than baryons. As an example, the interaction length for iron is about 132 g/cm^2 or 17 cm. The nuclear interaction length is always longer than the radiation length, i.e. $\lambda_{\text{int}}/X_0 > 1$ and the ratio scales with $Z^2A^{-2/3}$ and therefore increases with Z. This is why the electromagnetic calorimeter is always placed in front of the hadronic calorimeter in a collider experiment.

The nuclear interactions of high-energy hadrons interacting with a nucleus include mainly spallation, evaporation, and fission.

Nuclear spallation is the most likely interaction between the hadron and an atomic nucleus and is generally described as a two-stage process. The first stage is a fast intranuclear cascade: The hadron collides with the nucleons of the target nucleus, which in turn collide with other nucleons leading to many fast-traveling nucleons. This way particles like pions and other short-lived hadrons might also be generated. Some of these hadrons might be highly energetic enough to escape the nucleus. The second stage is *nuclear evaporation*: Particles such as free nucleons, α-particles, or larger nuclei evaporate leading to the de-excitation of the target nucleus. The evaporation continues until the excitation energy is less than the binding energy of one nucleon. Further energy might be released as photons. Instead of evaporation, *nuclear fission* might occur when highly energetic hadrons interact with very heavy target nuclei, such as uranium.

The nuclear binding energy needed to release the nucleons from the nucleus in a spallation event is "lost" in the sense that it cannot be measured with a calorimeter signal. Therefore, it is often called *invisible energy*. To a lesser extent, invisible energy can also stem from neutrinos produced in the interaction (also known as "escaped energy"). Another important source of invisible energy is the large number of neutrons evaporated during hadronic interactions. Such neutrons can deposit energy indirectly for example via elastic scattering or neutron

capture. These processes are usually delayed compared to electromagnetic interactions.

On average, invisible energy accounts for about 30–40% of the non-electromagnetic fraction of shower energy, i.e. the fraction of energy that is not carried away by π^0 or other electromagnetically interacting particles. Yet the amount of invisible energy fluctuates significantly from event to event leading a reduced precision with which hadron energies can be measured. Therefore, the energy resolution of electromagnetic calorimeters is commonly much better than of hadronic calorimeters.

2.1.4 Hadronic Showers

Similar to electromagnetic showers, once a high-energy hadron penetrates the material, a variety of processes can happen in a sequence leading to a hadronic shower. These processes can be either electromagnetic or hadronic in nature. Once a charged hadron enters the medium, it will ionize it and lose about 1–2 MeV $\mathrm{g}^{-1}\mathrm{cm}^{-1}$ of its energy (just like for example muons). After traveling on average one nuclear interaction length, a strong interaction likely occurs in which new hadrons (especially pions) are created as well as other nucleons and photons via spallation and subsequent evaporation. Small particles like mesons might travel about 25% farther than larger baryons before encountering a nucleus. As neutral hadrons do not ionize the medium, these nuclear interactions are their only way of losing kinetic energy.

The hadron shower develops similarly to an electromagnetic shower. The particles produced in the interactions, such as mesons, nucleons, and photons, can ionize the material and lead to further (nuclear) reactions thus inducing a cascade of secondary particles. Just like for electromagnetic showers, the longitudinal extent of hadron showers increases logarithmically with the incident hadrons' energy. However, unlike electromagnetic showers, the shower maximum does not depend on the indecent hadrons' energy. Instead, it is always located around $1\lambda_{\mathrm{int}}$ from the shower start.

This is because a hadron shower has two components: an electromagnetic component and a non-electromagnetic one (the hadronic component). This is schematically depicted in Figure 2.4. The electromagnetic component occurs due to the production of π^0 and η mesons in the initial reaction. Both decay almost instantaneously, with the π^0 lifetime of $\sim 10^{-17}$ s and η lifetime of $\sim 10^{-19}$ s. The π^0 decays into two photons and the η decays mostly into photons and π^0s. These photons can start an electromagnetic shower. The scale of this electromagnetic shower is governed by the radiation length X^0 which as discussed above is always smaller than λ_{int}.

The proportion of the hadron energy deposited via the electromagnetic component is called the *electromagnetic fraction* f_{em}. With increasing incident hadron energy, the number of particles produced in the intranuclear cascade increases. This leads to an increase in produced π^0 and η mesons and therefore to an increase of the electromagnetic fraction (since their energy does not induce more strong interactions). The average electromagnetic fraction scales with the particle energy E as [134]

$$\langle f_{\mathrm{em}} \rangle = 1 - \left(\frac{E}{E_0} \right)^{k-1} \tag{2.9}$$

where $k \sim 0.82$ is a constant and E_0 is a material-dependent parameter that depends on

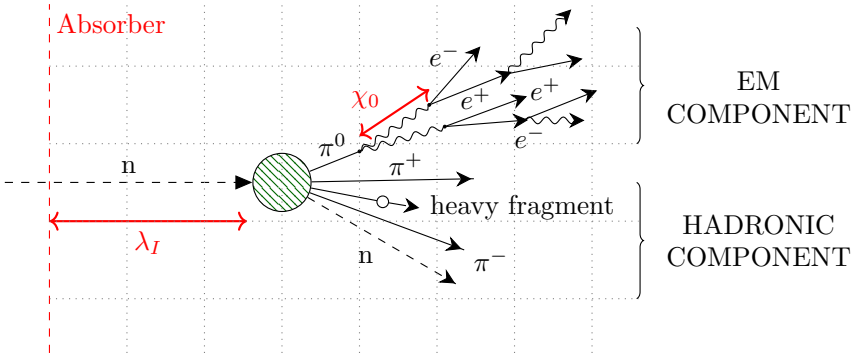


Figure 2.4: A hadron shower started by a neutron-nucleus interaction. The shower consists of an electromagnetic (em) component and a hadronic component. The hadronic component includes "invisible" energy due to nuclear binding energy and released neutrons. Figure taken from Reference [137].

the average multiplicity in the hadronic interaction (between 0.7–1.3 GeV in copper and lead for π-induced reactions, respectively). As an example, for a 10 GeV pion, the average electromagnetic fraction in copper is about $\langle f_{\text{em}} \rangle \sim 0.38$, and for a 1 TeV pion about $\langle f_{\text{em}} \rangle \sim 0.73$. However, for individual hadronic showers, the actual electromagnetic fraction can fluctuate significantly.

The hadronic component of the shower involves all the nuclear interactions discussed in Section 2.1.3. This includes the invisible energy which is another part of the shower that varies strongly from shower to shower. While the electromagnetic component develops promptly due to particles generated at relativistic energies, the hadronic component can be delayed due to nuclear excitations ($\sim \mu$s) and thermal neutrons ($\sim \mu$s to ms). Thermal neutrons can travel long distances in the material before being absorbed and deposit energy in quite a different location than the initial hadron.

2.2 Calorimeters

Calorimeters are devices that absorb incident particles with the purpose of measuring their energy. They are made up of one or multiple absorption media together and use various sensors to read out signals used for the calculation of the deposited energy. As the longitudinal shower depth scales logarithmically with the incident particles' energy, most of the energy of the particles reaching the calorimeter can indeed be measured with a moderate amount of material. Every particle detector at collider experiments uses calorimeters to measure and reconstruct in detail particles produced in the collision. Typically, in a detector experiment, one distinguishes the *electromagnetic calorimeter* (ECAL) and the *hadronic calorimeter* (HCAL).

ECALs are optimized for measuring the energy of electromagnetically interacting particles such as electrons, protons, and photons. HCALs on the other hand are optimized for measuring hadron energies of particles such as protons, neutrons, charged and neutral pions and other mesons. The design choices made for either apparatus relate to the nature of the

interactions discussed in the previous section, e.g. the HCAL is usually significantly thicker than the ECAL and sits behind it as the nuclear interaction length is much larger than the radiation length. Note, however, that it is still very much possible for hadrons to deposit energy in the ECAL and vice versa as all these interactions are highly stochastic.

There are various subtleties in the design of calorimeters, how they respond to different particles, and how their energy resolution is optimized.

2.2.1 Calorimeter Configuration

The type of calorimeter design is generally separated into two categories: *homogenous calorimeters* and *sampling calorimeters*.

Homogenous Calorimeters

Homogenous calorimeters are built entirely of 'active material', i.e. material in which particles can induce a measurable signal. As all energy deposited in the calorimeter is measured, the energy resolution can be very good. Typical active materials are crystals that produce scintillation light (e.g. lead tungstate ($PbWO_4$)) or Cherenkov light (e.g. lead glass). Another less common material for a homogenous calorimeter is based on liquified noble gases, which can produce very bright scintillation light in the ultraviolet wavelength region. Liquid argon can also be used, although here ionization charges produce the signal, not scintillation light.

An example of a homogenous calorimeter used in a collider experiment is the ECAL in the CMS detector which uses a total of 75,848 lead tungstate crystal scintillators. Having a short radiation length of $X_0 = 0.89$ cm and a Molière radius of $\rho_M = 2.2$ cm allows for a compact design as the crystal length of 230 mm equates to 25.8 X_0 [98]. Arguably the largest homogenous calorimeters in use are neutrino detectors which use the production of Cherenkov light in water. Super-Kamiokande [138] uses a detector with 50,220 tons of ultrapure water and IceCube [139] utilizes 1 km^3 of ice for detection.

Sampling Calorimeters

Sampling calorimeters use both active and 'passive materials' in their construction. The passive material is used to induce interactions with the incident hadron to produce secondary particles that are then measured in the active material. Typically, the passive absorber medium is a dense material: Lead or tungsten are often used in ECALs and iron, steel, or copper in HCALs. The higher material density in the passive medium than in the active allows for a more compact design of sampling calorimeter compared to homogenous ones. It can also save costs with the passive material usually being cheaper than the more complex active medium. Just like in homogenous calorimeters, the active materials generate a signal in the form of scintillation light, Cherenkov light, ionization, or via an electron-hole pair in a semiconductor.

Most commonly, sampling calorimeters are implemented with active and passive layers interleaved in a "sandwich" structure with the layers orientated such that the particle passes them perpendicularly in an alternating fashion. While this structure works fine for many experiments, it might not necessarily be the optimal arrangement for 4π detectors at collider

experiments. Other options include placing the scintillators oriented perpendicular to the beam axis (as is done in the ATLAS HCAL) or using an "accordion" structure, where a liquid argon active medium is contained in lead absorbers (like in the ATLAS ECAL). Additionally, longitudinal and lateral segmentation of the active material into small cells can provide granular spatial information about a particle's path. This can enable advanced reconstruction methods for improving the detectors' energy resolution (see Section 2.3).

2.2.2 Calorimeter Response

To accurately measure the energy of incoming particles, one needs to properly understand how the calorimeter responds to deposited energy and how a signal is generated. The calorimeter response is defined as the ratio of the average calorimeter signal to a unit of deposited energy [133]. In a scintillation crystal as an active medium, this equates to the number of photons per deposited energy in eV. The *linearity* of the calorimeter measures the proportionality of the measured signal to the energy of the incident particle. In an ideal linear calorimeter, the calorimeter response is constant. The measurement precision the calorimeter can achieve is given by the *energy resolution*, which is defined as the relative width of the signal distribution $\frac{\sigma}{E}$. As the energy depositions of electromagnetic and hadronic showers are governed by different processes, the calorimeter's response to either shower type is subsequently discussed separately.

Electromagnetic Response

Electromagnetic showers deposit energy with several well-understood and relatively simple processes allowing for the construction of ECALs with a high degree of linearity. In particular homogenous calorimeters are well suited for measuring such showers with a high resolution as most of the energy of the incident particle is used to create a measurable signal. Deviations from a completely linear response might still occur due to experimental constraints such as saturation effects in the photodetectors and read-out electronics or due to shower leakage, i.e. energy depositions outside the calorimeter medium.

Every shower development is a stochastic process and therefore shower-to-shower functions are to be expected. Such fluctuations follow Poissonian statistics. Overall, the created shower particles N_{shower} are proportional to the incident particle energy E. Further, the measured signal of the calorimeter is proportional to the number of visible particles N_{vis} that create said signal. Hence, the higher the energy of the incident particle, the higher the possible precision of measurements. For a homogenous calorimeter, we expect $N_{\text{vis}} \approx N_{\text{shower}}$. Therefore, the stochastic fluctuations of N_{shower}, the *intrinsic shower fluctuation*, directly relate to the energy resolution $\frac{\sigma}{E}$ as the width of the signal distribution scales as $\sigma \propto \sqrt{N_{\text{shower}}} \propto \sqrt{E}$. These intrinsic shower functions are dependent on the particle type, meaning the calorimeter resolution depends on a specific particle such as electrons, pions (hadrons), or muons.

In sampling calorimeters, only a fraction of the produced particles and the deposited energy produce a signal, since a large fraction is absorbed in the passive material ($N_{\text{shower}} >> N_{\text{vis}}$). For these calorimeters N_{samp} gives the number of particles in the active layers. It is given by $N_{\text{samp}} = f_{\text{samp}} \cdot N_{\text{shower}}$, where f_{samp} denotes the *sampling fraction* defined as the ratio of

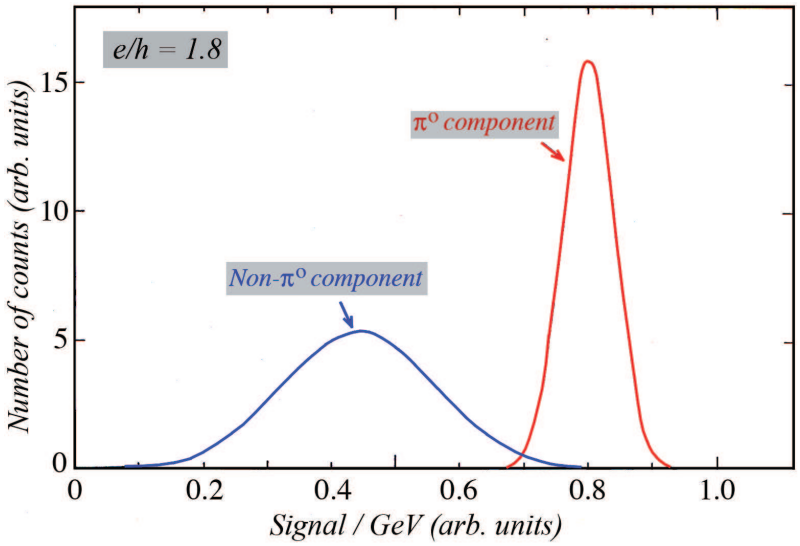


Figure 2.5: Schematic representation of the response functions for the electromagnetic (ϕ^0) and the hadronic (non-ϕ^0) component of a hadronic shower in an under-compensating calorimeter. The ratio of the mean of the distributions yields $e/h = 1.8$. Figure taken from Reference [133].

visible energy E_{vis} deposited in the active material to the total absorbed shower energy E_{shower} of minimum ionizing particles (MIPs). Typically, the sampling fraction can be characterized using the calorimeter response to MIPs as

$$f_{\text{samp}} = \frac{E_{\text{MIP}}^{\text{active}}}{E_{\text{MIP}}^{\text{active}} + E_{\text{MIP}}^{\text{passive}}} \tag{2.10}$$

where $E_{\text{MIP}}^{\text{active}}$ and $E_{\text{MIP}}^{\text{passive}}$ is the energy loss of a MIP in the active and passive material, respectively. The measurement of the sampling fraction is usually done using muons as they are the closest to a MIP provided by nature.

The stochastic fluctuations of N_{samp} are called *sampling fluctuations*. These fluctuations depend on the sampling fraction as well as on the sampling frequency, i.e. the number of alternating active and passive sampling elements. Sampling fluctuations usually dominate other sources of statistical uncertainties in sampling calorimeters such as shower leakage, instrumental effects (e.g. electronic noise) or signal quantum fluctuations (e.g. photoelectron statistics). As the sampling fluctuations follow Poisson statistics their resulting impact on the energy uncertainty can be written as $\sigma_{\text{samp}} = \sqrt{d/f_{\text{samp}} \cdot E}$, where d is the thickness of an active layer. [135]

Hadronic Response

The response of a calorimeter to hadronic showers is more complex than to electromagnetic showers since hadronic showers contain a well-measurable electromagnetic component and a more difficult hadronic component which also includes unmeasurable invisible energy.

Generally, the calorimeter response to a hadronic shower Φ can be expressed as

$$\Phi = f_{\text{em}} \cdot e + (1 - f_{\text{em}}) \cdot h \tag{2.11}$$

where f_{em} is the electromagnetic fraction of the shower, e is the response to the electromagnetic component, and h is the response to the hadronic component. The calorimeter response to Φ is non-linear because f_{em} increases with the incident hadron energy and due to the invisible energy phenomenon.

In general, a hadronic calorimeter has $h < e$ due to parts of the hadronic energy being 'invisible'. Such a calorimeter would be called *non-compensating* or *under-compensating* as $e/h > 1$. The response functions for such an under-compensating calorimeter are shown in Figure 2.5. If $e = h$ the calorimeter is called *compensating*. A compensating calorimeter has an equal response to the electromagnetic and the hadronic fraction of the hadron shower, allowing for a linear calorimeter with an improved energy resolution. The ratio $e/h < 1$ is called *over-compensating*. Typically, calorimeters are undercompensating with e/h values between 1.5–2.0.

With sampling calorimeters, it is possible to build a compensating calorimeter by carefully optimizing the active and passive materials and thicknesses to either lower the electromagnetic response or to raise the hadronic one. Famously the ZEUS hadronic calorimeter at the HERA collider achieved this by using uranium absorbers and plastic scintillators. Originally, the hope was to offset the invisible energy loss through nuclear binding energy with the neutrons produced in the uranium fission, but it turned out that these are uncorrelated and in hindsight, lead would have been an even better absorber material. ZEUS still achieved $e/h \sim 1$ and a record-breaking energy resolution for pions of $\sigma/E = 35\%/\sqrt{E} + 2\%$ [140].

If it is not experimentally useful to build a compensating calorimeter, another option is to try to measure the electromagnetic and hadronic components on a shower by shower basis and perform a recalibration for each individual shower. This can be achieved with a *dual-readout calorimeter* [141] which measures both scintillation light and Cherenkov light in the active material. As the Cherenkov signal is mainly produced by the electromagnetic fraction of a shower, it can be used for particle identification and the proportion of Cherenkov light compared to the scintillator signal is applied to calculate the electromagnetic fraction f_{em} of a hadronic shower on a per-shower basis. This can be used as a compensation factor to reach a very high overall energy resolution.

Another option is to increase the *spatial resolution* of the calorimeter such that it becomes possible to identify the electromagnetic sub-showers triggered by π^0 and η mesons within the hadronic shower. Once identified, the local energy depositions can be reweighted. This technique is known as *software compensation* and has been demonstrated at the $H1$ experiment using local hit energy densities for the compensation [142]. A high spatial resolution is also needed for the particle flow approach to calorimetry discussed below in Section 2.3.

2.2.3 Energy resolution

The energy resolution $\frac{\sigma}{E}$ of a calorimeter is parameterized as

$$\frac{\sigma}{E} = \sqrt{\left(\frac{a}{\sqrt{E}}\right)^2 + b^2 + \left(\frac{c}{E}\right)^2} = \frac{a}{\sqrt{E}} \oplus b \oplus \frac{c}{E} \tag{2.12}$$

where $\oplus$ denotes the addition in quadrature, a, b, and c are free parameters usually quoted in percent, and E the particle energy in GeV. The first term is the *stochastic term*, the second term is the *constant term*, and the third term is the *noise term*. They are added in quadrature because the sources for each term are uncorrelated. Their motivations are:

(a) The stochastic term originates from the intrinsic shower fluctuations and the sampling fluctuations which are energy-dependent and follow Poissonian statistics. For ECALs a is of the order of 10% and for non-compensating HCALs of the order of 60%. The best value ever was achieved with the HCAL of the ZEUS detector at the HERA collider with $a = 35\%$ [140].

(b) The constant term is due to instrumental effects, non-uniformities in the detector such as dead regions, and calibration uncertainties. These effects scale with the energy and therefore the term is dominant at high energies with b in the order of a few percent.

(c) The noise term describes the measurement uncertainty because of electronic noise in the read-out. This term is dominant at low energies. Sometimes the noise term is neglected and one sets $c = 0\%$ yielding the simplified form

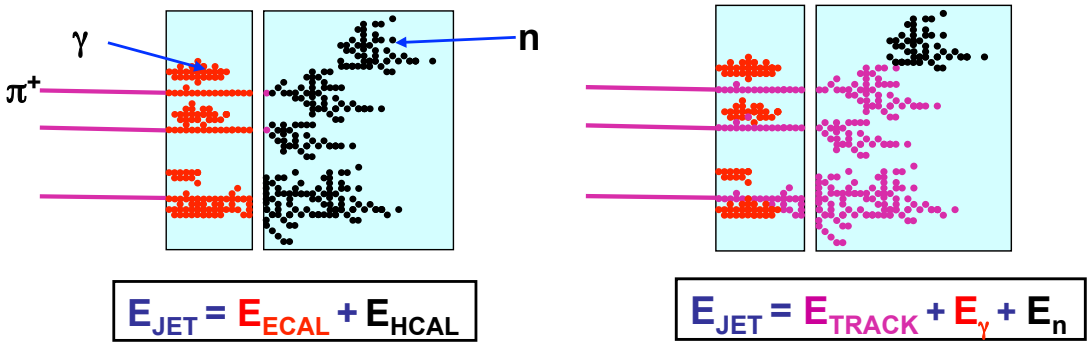
$$\frac{\sigma}{E} = \frac{a}{\sqrt{E}} \oplus b. \tag{2.13}$$

The energy resolution is usually measured with particles of known energy in a test beam setup. The results are then fitted with the above function to extract the parameters a, b, and c.

2.3 Particle Flow Calorimetry

To measure the energy of a single particle or a whole jet with a calorimeter, traditionally all energy depositions in the ECAL and HCAL are summed up. Using a typical HCAL, this results in a jet energy resolution on the order of $\sigma/E \approx 60\%/\sqrt{E}$. However, for the physics program at a future lepton collider like the ILC a jet energy resolution of 3–4% in the range of 45–250 GeV is targeted [24]. This equates to a calorimeter resolution of about $\sigma/E \sim 30\%/\sqrt{E}$. Such a high jet resolution is necessary to cleanly separate $W^\pm$ and Z^0 boson decays into multi-jet final states. This separation requires an invariant mass resolution on the order of the gauge boson widths, i.e. $\sigma/m = 2.7\% \approx \Gamma_{W^\pm}/m_{W^\pm} \approx \Gamma_{Z^0}/m_{Z^0}$, enabling a 3.6σ separation of the $W^\pm \to q\bar{q}$ and $Z^0 \to q\bar{q}$ mass peaks [25].

This high requirement on the jet energy resolution motivates the usage of the *particle flow* approach to calorimetry in future collider experiments. The purpose of *Particle Flow Algorithms* (PFA) is the reconstruction of the individual jet contributions for increased



(a) Traditional calorimetry **(b)** Particle flow calorimetry

Figure 2.6: Comparison between the traditional and the particle flow approach to calorimetry: **(a)** Traditionally a jet is reconstructed by combining the ECAL and HCAL response. **(b)** In particle flow calorimetry the jet contributions are segmented and the jet is reconstructed using the optimal detector sub-system for a given particle, i.e. the tracker information for charged particles, the ECAL for photons, and the HCAL for neutral hadrons such as neutrons. Figures taken from Reference [143].

measurement precision. The reconstruction paradigm shifts from reconstructing the overall energy deposition in the calorimeter to the reconstruction of individual particles. PFA is enabled by the possibility of fine spatial segmentation (high granularity) of modern calorimeters and advanced software tools for clustering and reconstruction.

At LEP it was shown that on average a jet deposits about 60% of its energy via charged hadrons, 30% via photons, and 10% via neutral hadrons. [25] Therefore, with a traditional combination of ECAL and HCAL, 70% of the jet energy is measured with the subpar resolution of an HCAL with $\sim 60\%/\sqrt{E}$, leading to an overall poor precision in jet measurements. With the particle flow approach, the most efficient detector sub-system for each particle type is used. For the measurement of charged particles, this is actually the tracking system. Photons can be measured well in the ECAL and the HCAL is only used for measuring the energy of neutral hadrons. This reduces the HCALs' usage to the measurement of only 10% of the jet energy deposition, improving the overall jet resolution. To gain this performance, an excellent tracking system is needed in combination with calorimeter systems that have a fine longitudinal and lateral segmentation. This is necessary to be able to distinguish the energy depositions of the different particles to choose which sub-system shall be used for the energy reconstruction. The separation of particles and the clustering of energy deposits is done with software tools such as *PandoraPFA*. [25]

It is expected that at the ILC experiments, particle flow will lead to a jet energy resolution of 3–4% in the energy range 45–250 GeV, which equates to about $30\%/\sqrt{E}$. With particle flow, the jet energy resolution can be parameterized as

$$\sigma_{\text{jet}} = f_{\text{charged}} \cdot \sigma_{\text{track}} \oplus f_{\gamma} \cdot \sigma_{\text{ECAL}} \oplus f_{\text{neutral}} \cdot \sigma_{\text{HCAL}} \oplus \sigma_{\text{confusion}} \oplus \sigma_{\text{leak}} \tag{2.14}$$

where f_{charged}, f_{γ}, and f_{neutral} are the fractions of the jet energy deposited by charged particles, photons, and neutral hadrons, respectively. The parameters σ_{track}, σ_{ECAL}, and σ_{HCAL} are

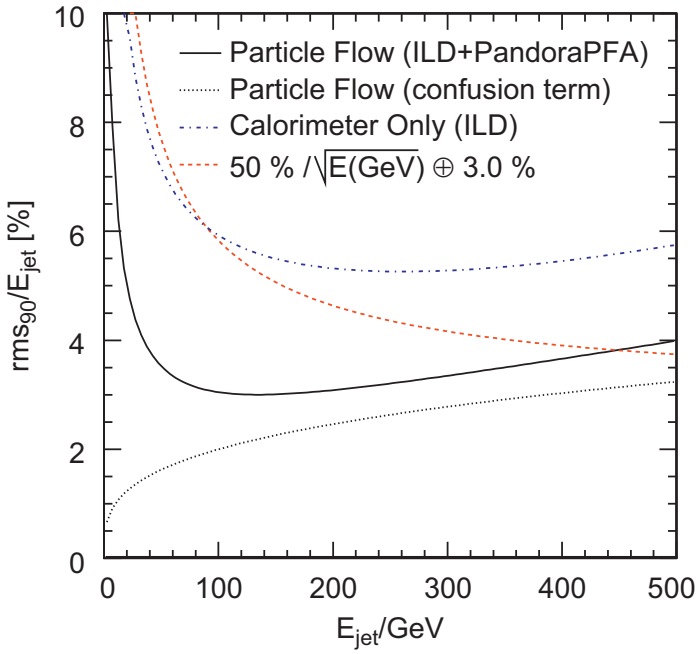


Figure 2.7: Estimated jet energy resolution achieved with the particle flow approach at the ILD compared to using only the calorimeter. Additionally, the confusion term that scales with the energy is plotted. Figure taken from Reference [25].

the energy resolutions of the tracking system, the ECAL, and the HCAL, respectively. The term $\sigma_{\text{confusion}}$ is arising from the mis-assignment of energy depositions or tracks and σ_{leak} is due to particle leakage outside the instrument area.

The *confusion term* $\sigma_{\text{confusion}}$ is a limiting factor of the particle flow approach, especially at higher energies. There are three main sources of confusion:

- A charged hadron might be identified as a neutral hadron. This leads to double counting of its energy contribution since both the track information and the HCAL measurement will be used for reconstruction.
- A neutral hadron might be identified as a charged hadron. This leads to a loss of energy, as its HCAL component will be discarded.
- Photons close to neutral hadrons might be counted together as a neutral hadron contribution leading to a loss of photon energy.

The confusion term scales with the energy as more particles are produced at higher energies making it more difficult to distinguish them. The estimated jet energy resolution that could be obtained at the ILD, as well as the confusion term is shown as a function of jet energy in Figure 2.7.

Particle flow is an improved version of energy flow that was used in the past at collider experiments such as D0, H1, and OPAL to improve jet energy resolutions. Currently, CMS is using particle flow to improve the detector performance [144]. However, none of these detectors were specifically designed with particle flow reconstruction in mind. At future

colliders, the experiments are proposed to feature excellent tracking systems and highly granular calorimeters to fully exploit the particle flow approach. Higher granular calorimeters enabling more precise measurements increase the need for high fidelity fast simulation techniques, the topic of this thesis.

2.4 Calorimeter Simulation

Simulating particles traversing the calorimeters is the most computationally intensive part of a detector simulation. *Full simulation* tools rely on physics-based Monte Carlo (MC) techniques that track each particle in the calorimeter medium and simulate all possible interactions. Considering the amount of simulation needed for future collider experiments, only relying on such physics-based simulations is not sustainable. Therefore, various methods for computationally efficient *fast simulation* exist that alter or bypass parts of the simulation chain. The generative machine learning models developed in the context of this thesis are examples of such fast simulations, but there are also several methods for fast simulation that do not use machine learning. In the following, the full MC-based simulations as well as the fast alternatives are discussed.

2.4.1 Monte Carlo Simulation

GEANT4 (Geometry and Tracking) [118, 145] is a transport software toolkit for the simulation of particle-matter interactions. It uses various physics models to perform a first principle tracking of the paths and interactions of individual particles over a wide energy range. In particle physics, GEANT4 is synonymous with full MC detector simulations. Therefore, MC calorimeter simulations are discussed here only in the context of GEANT4.

As mentioned in Section 1.5, a transport software like GEANT4 takes as input a detector description as well as individual particles or hadronized showers, e.g. from an event generator. The detector description contains a "hierarchy" of many volumes, each made up of a specific compound. The interactions between the particles and the detector are then modeled as a step-by-step track with the transport software. At each of these simulated steps within a track (or when crossing the boundary between volumes), a specific interaction is chosen out of all the possible options.

These interaction options are characterized by the mean free path of a given process λ. In a composite material it is defined as the inverse of the macroscopic cross-section:

$$\lambda(E) = \left(\sum_i [n_i \cdot \sigma(Z_i, E)] \right)^{-1} \tag{2.15}$$

where n_i is the number density of the i-th element in the material and $\sigma(Z, E)$ is the total cross-section per atom of the given interaction. How many mean free path length λ a particle travels between x_1 and x_2 is given by

$$n_\lambda = \int_{x_1}^{x_2} \frac{\mathrm{d}x}{\lambda(x)}. \tag{2.16}$$

The probability that an interaction occurs within the distance x is then given by

$$c(x) = 1 - p(x) = 1 - e^{-n_\lambda}. \tag{2.17}$$

The total step size s can now be expressed as

$$s(x) = n_\lambda \cdot \lambda(x) \tag{2.18}$$

where n_λ can be sampled at the start of the computation with $n_\lambda = -\log(\eta)$ and η sampled from a uniform distribution. This way s is calculated for all possible processes, the shortest s is selected and the simulation performs that specific process. The simulation continues then with the next step and further processes until the particle energy falls below a threshold and its energy is deposited or until the particle leaves the detector volume. Note that at a material boundary determined in the detector volume, a process occurs regardless of the step size. For complex detectors, this may reduce the mean step size and lead to overall more steps taken and processes simulated. As particle are individually tracked, GEANT4 is a very accurate, albeit computationally expensive simulator.

GEANT4 uses so-called *physics lists* to define the possible processes and their cross-sections for specific particle types. The standard electromagnetic package in GEANT4 models the interactions of charged particles above 1 keV well. However, hadronic interactions are much more complex and for specific use-cases multiple physics lists including different combinations of hadronic models are offered. For the simulation of photons in the ECAL, the popular physics list QGSP–BERT is used in this thesis. This physics list includes the standard EM processes, as well as the Quark-Gluon string pre-compound Model [146] for high energies and the Bertini Cascade Model [147] for low hadron energies. For the energies in between where neither model is optimal, an empirical parameterization is applied.

2.4.2 Fast Simulation

Full MC transport simulations like GEANT4 are computationally expensive as every single particle needs to be tracked individually. Further, these simulations cannot be easily parallelized as each step of the simulations depends on the previous one. Therefore, multiple fast simulation (FastSim) techniques have been developed to accelerate the production of simulated events. These usually compromise fidelity for speed, but are necessary to achieve the needed amount of simulated events for the physics programs at large collider experiment and it is standard practice to use fast simulation wherever the full simulation is not needed. Traditional fast simulation approaches include methods and tools such as:

- **Detector Parameterization:** A common framework for the fast simulation of the general-purpose collider detector is DELPHES [148]. It takes the event generator output and essentially smears the momenta of the long-lived visible particles that would reach the detector sub-systems like the calorimeters. The smearing applied is determined by the detector resolution and also takes into account the impact of the magnetic field. As the calorimeter resolution needs to be known a priori, DELPHES cannot be used to simulate novel systems. It also cannot be plugged seamlessly into the real experiments'

reconstruction chain no calorimeter hits are produced like with GEANT4 because there is no explicit detector geometry.

- **Shower Parameterization:** The development of electromagnetic showers is well understood and their spatial energy distribution can be parameterized relatively simply as [149, 150]

$$dE(\vec{r}) = E\, f(t)dt\, f(r)dr\, f(\theta)d\theta \tag{2.19}$$

where $f(t)$, $f(r)$, and $f(\theta)$ are longitudinal, radial, and azimuthal energy distributions, respectively. These distributions are determined by the shower depth in radiation lengths t, the radial distance in Molière radii r, and the polar angle θ. This parameterization can be adjusted to also include hadronic showers, in particular by considering the fluctuations in the f_{em} fluctuations. It can also be adapted to apply for homogenous and sampling calorimeters (by accounting for sampling fluctuations). Within GEANT4, a fast shower simulation based on these parameterizations dubbed GFLASH is implemented. In a recent thesis [151], it was shown that it is challenging to apply GFLASH for highly-granular calorimeters and that it can be outperformed by a simple generative model.

- **Shower Library:** A shower library, also known as *frozen showers* [152], consists of a large set of pre-simulated showers of different particle types, energies and orientations. During the simulation, if particles with specific conditions are detected (such as a certain energy), the simulation of said particle is aborted and instead substituted with a corresponding shower chosen from the library. At the ATLAS experiment, this technique has been used for the fast simulation of low-energy electromagnetic showers.

A novel approach to fast simulation is the use of generative machine learning models. These large parametric models, often with millions of parameters, promise to be able to simulate the detector response in much greater detail than traditional fast simulation methods while maintaining a significant speed-up compared to full MC simulations. The advantage of generative models over MC methods is that they do not track individual particles and are therefore easily parallelizable. However, they rely still on training data usually produced by a full simulation like GEANT4, but interestingly could also be trained on real data [153]. In recent years, many generative models have been developed for calorimeter shower simulation which achieve ever-increasing fidelity as well as computational efficiency. This innovation is largely driven by novel approaches to generative modeling using deep neural networks as well as the increased availability of large-scale computing resources such as high-performance GPUs.

The early seminal work in exploring generative models for calorimeter simulation was CALOGAN [39–41], a generative adversarial network (GAN) trained to generate electromagnetic showers as images with a total of 504 pixels. Since then, various different types of generative modeling paradigms have been explored for fast calorimeter simulation, including GANs [39–48], autoencoder-variants [1, 6, 9, 49–51], normalizing flows [52–58], and diffusion models [3, 5, 59–61]. This list is by no means exhaustive; a living review of the field is maintained at Reference [26]. In the recently finished community challenge, dubbed

"CaloChallenge", various state-of-the-art models were compared, and the results are about to be published in Reference [14].

The models are fast progressing and achieving every increasing accuracy even on highly-granular calorimeters with $\mathcal{O}(10^4)$ readout channels. At the LHC, the ATLAS collaboration has recently started using GANs in their fast simulation framework [154]. Fast progress is made of efficiently incorporating these generative models even in the full simulation chain parallel to GEANT4 [13, 155]. Adjacent work has shown that a generative model can indeed amplify the statistics of a given type of event [156–158].

When evaluating these models in terms of simulation speed, it is important to consider the cost of the hardware, i.e. traditional fast simulation methods can run on a CPU, while many generative models require more expensive GPUs. Therefore, in this thesis, we compare the acceleration of the calorimeter simulation process on both CPU and GPU hardware — with the same type of CPU is used for the GEANT4 simulation and benchmarking the generative models. The speed-up and the fidelity achieved by the generative models studied in the context of this thesis are discussed in Chapter 5 and Chapter 6.

Chapter 3

Machine Learning

Machine learning is a subfield of artificial intelligence (AI) and generally refers to the development of predictive models that are automatically optimized on data, i.e. they "learn" from data. The field of machine learning has seen rapid developments in recent years due to the availability of large datasets, novel algorithms, and a stark increase in computational power, especially in the form of graphical processing units (GPUs).

Particle physicists have been using machine learning techniques for decades to analyze data collected at particle colliders. The most frequent types of models are based on decision trees and neural networks. They are used for various applications such as particle identification, event reconstruction, anomaly detection, and fast simulation. The rapid development of machine learning has led to an increased adoption of trainable models in particle physics analyses as physicists are aiming to extract more information from the available data. A living review of the machine learning for particle physics research can be found in Reference [26].

An extensive introduction to Machine Learning and Deep Learning is given in References [159, 160]. For an introduction targeted specifically towards physicists References [161, 162] can be recommended. The discussion of model training, deep neural networks, and convolutional neural networks in this chapter follows Reference [161].

In this chapter, an overview of the most important machine learning concepts used in this thesis is provided. The optimization, or "training", of predictive models are discussed in Section 3.1. Deep neural networks as the general class of models used in this thesis are introduced in Section 3.2 and their training is discussed in Section 3.3. Finally, two specific types of neural network architectures used in this thesis are outlined: convolutional neural networks (CNNs) in Section 3.4 and graph- and set-based neural networks in Section 3.5.

3.1 Model Optimization

There are generally three ingredients needed to tackle a problem with machine learning: a dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=0}^{N}$ with N entries, a model $\mathbf{g}_{\boldsymbol{\theta}}$ with trainable parameters $\boldsymbol{\theta}$, and a differentiable *loss function* $L(\mathbf{X}, \mathbf{g}_{\boldsymbol{\theta}})$ which quantifies how well the model $g_{\boldsymbol{\theta}}$ is performing for a given task on the dataset $\mathbf{X}$. The loss function is also often known as "error function", "cost function", or "value function" and although technically different, for practical purposes

the terms are used interchangeably in this thesis. To make the model perform well on a given task, the parameters $\boldsymbol{\theta}$ are optimized by minimizing the loss function. This optimization process is known as "training" the model. Conceptually, the training process is very similar to 'fitting' a function to a dataset via χ^2 minimization as done regularly in particle physics analyses.

The training process is typically done with an algorithm based on *gradient descent*. The idea is to iteratively update the parameters $\boldsymbol{\theta}$ in the direction of the negative gradient of the loss function. In the optimal case, a minimum of the loss function is found after many update steps and the model is said to be "converged" meaning the training is finished. In practice the optimization process is very difficult as modern models contain millions of parameters, are trained on millions of samples, and the loss functions are often highly non-linear and non-convex with many local minima in a high-dimensional parameter space. Hence, advanced optimization algorithms are needed to train these models efficiently. These algorithms are often referred to as *optimizers* and are based on the general of idea of gradient descent.

3.1.1 Gradient Descent

At the start of the model training, the parameters $\boldsymbol{\theta}$ are randomly initialized as $\boldsymbol{\theta}_0$. Since the loss function is minimized with respect to the parameters $\boldsymbol{\theta}$, we denote the loss in the following as $L(\boldsymbol{\theta})$. With the simplest gradient descent (GD) algorithm, the parameters are updated iteratively as

$$\mathbf{v}_t = \eta_t \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) \quad \text{and} \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t \tag{3.1}$$

where the scale of the optimization at time step t is controlled by the *learning rate* η_t. Usually, the learning rate is a set parameter and allows the model to converge to a *local minimum*. However, the smaller the learning rate, the slower and the more computationally expensive the optimization process becomes. On the other hand, a too large learning rate can lead to the model not converging. A well-chosen learning rate – either fixed or with specific time-dependent scheduler – is a crucial part of model development.

The simple GD algorithm has several drawbacks. It is deterministic and therefore converges to a local minimum of the loss function based on the initialization $\boldsymbol{\theta}_0$. Since the loss landscape is often very rugged, this minima might not be close to the *global minimum* which might provide significantly better predictive performance. Computing the gradients with GD is also very computationally expensive, since the loss is computed as the mean over all N training data samples. Additionally, the GD algorithm is very sensitive to the choice of the learning rate and can be slow to converge in high-dimensional parameter spaces. To address these drawbacks, several variants of GD have been introduced which are widely adopted in practice.

3.1.2 Stochastic Gradient Descent

A widely used variant of GD is *stochastic gradient descent* (SGD) [163, 164]. In SGD, the gradient is approximated by as small subset of the training data which introduces a stochastic element to the optimization. This subset is called a *minibatch* and usually contains $\mathcal{O}(10 - 1000)$ randomly selected samples. If the minibatch contains n samples, the whole

training dataset is divided into N/n minibatches. After the algorithm has iterated over all minibatches, the training set is shuffled again and split into new minibatches. One full iteration over all minibatches is called an *epoch*. The SGD algorithm can then be expressed as

$$\mathbf{v}_t = \eta_t \nabla_{\boldsymbol{\theta}} L_{\mathrm{MB}}(\boldsymbol{\theta}_t) \quad \text{and} \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t \tag{3.2}$$

where $L_{\mathrm{MB}}(\boldsymbol{\theta}_t)$ is the loss function computed on a minibatch. SGD addresses two shortcomings of GD mentioned before: It adds stochasticity to the gradient calculation which helps the optimization to not get stuck in a local minimum, and it is computationally more efficient than calculating the gradients on the whole dataset.

3.1.3 Momentum

To impose an overall direction to the descent in gradient space, SGD is usually used in conjunction with *momentum*. With the introduction of a momentum parameter γ, the optimization is modified to *gradient descent with momentum* (GDM) given by

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \eta_t \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) \quad \text{and} \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t \tag{3.3}$$

where γ is a parameter between 0 and 1 and the minibatch subscript is omitted for simplicity. Since $\mathbf{v}_t$ is a running average of the gradients, $(1-\gamma)^{-1}$ sets a time horizon over how many recent gradient updates the average is taken. Setting $\gamma = 0$ yields just the standard SGD algorithm. A momentum parameter $\gamma > 0$ helps to accelerate the optimization process in directions where the gradient points often, while suppressing it in high-curvature directions. The parameter especially helps in the beginning of the training, where the gradient landscape is particular noisy.

3.1.4 Adaptive Learning Rate

Ideally, the learning rate should change depending on the current gradient landscape. It should be large in flat regions to speed up the optimization and small in steep regions to avoid missing the minimum. One option would be to apply Newton's method and use the inverse of the Hessian $H(\boldsymbol{\theta}_t)$ to scale the gradient. This results in the update rule for Newton's method as

$$\mathbf{v}_t = H^{-1}(\boldsymbol{\theta}_t) \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) \quad \text{and} \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{v}_t. \tag{3.4}$$

However, calculating the approximate Hessian is computationally expensive and often not possible for large models. The alternative was introduced with optimizers that track the second momentum of the gradients, such as the RMSprop (Root Mean Square Propagation) [165] and the Adam (Adaptive Moment Estimation) [166] optimizers.

Considering the second moment of the gradients $\mathbf{s}_t = \mathbb{E}[\mathbf{g}_t^2]$, the optimization rule for RMSprop is given by

$$\begin{aligned} \mathbf{g}_t &= \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) \\ \mathbf{s}_t &= \beta \mathbf{s}_{t-1} + (1-\beta)\mathbf{g}_t^2 \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta_t \frac{\mathbf{g}_t}{\sqrt{\mathbf{s}_t} + \epsilon} \end{aligned} \tag{3.5}$$

where β is time over which the second momentum is averaged and ϵ is a small constant to avoid division by zero. Typically, these parameters (also known as hyperparameters since they are set before the training and not learned) are set to $\beta = 0.9$, $\eta_t = 10^{-3}$, and $\epsilon = 10^{-7}$. Due to dividing the learning rate by the square root of the second momentum, the learning rate is increased when the gradient was small over many previous steps, i.e. in a flat gradient landscape, and vice versa. This way the optimization process can be accelerated significantly.

The ADAM optimizer adaptively changes the learning rate based on both the first and the second moment of the gradients, i.e. $\mathbf{m}_t = \mathbb{E}[\mathbf{g}_t]$ and $\mathbf{s}_t = \mathbb{E}[\mathbf{g}_t^2]$, respectively. Additionally, a bias correction to the running averages is performed. The update rule for ADAM is given by

$$\begin{aligned}
\mathbf{g}_t &= \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) \\
\mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \\
\mathbf{s}_t &= \beta_2 \mathbf{s}_{t-1} + (1-\beta_2)\mathbf{g}_t^2 \\
\hat{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1-\beta_1^t} \\
\hat{\mathbf{s}}_t &= \frac{\mathbf{s}_t}{1-\beta_2^t} \\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta_t \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon}
\end{aligned} \tag{3.6}$$

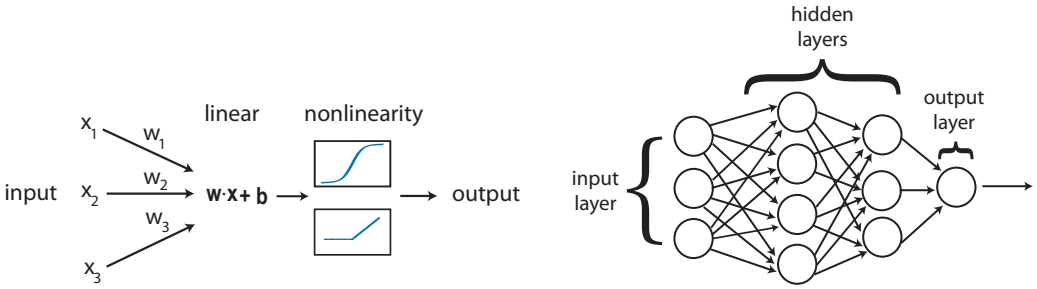
where β_1 and β_2 are parameters that control the averaging time of the first and second momentum, respectively. Typically, the hyperparameters are set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\eta_t = 10^{-3}$, and $\epsilon = 10^{-7}$.

The update expression can be rewritten in terms of the variance $\boldsymbol{\sigma}_t^2 = \hat{\mathbf{s}}_t - \hat{\mathbf{m}}_t^2$. For a single parameter θ_t it is given by

$$\Delta\theta_{t+1} = -\eta_t \frac{\hat{m}_t}{\sqrt{\hat{\sigma}_t^2 + \epsilon}}. \tag{3.7}$$

For past gradient updates with a low variance, i.e. when the gradient landscape is consistent, the update approximates the learning rate as $\Delta\theta_{t+1} \approx -\eta_t$. This has the same effect as cutting off large gradients, i.e. if a large step shall be taken, although the overall gradient landscape is flat, the optimizer scales that step down. In the limit of high variance, i.e. the gradient landscape fluctuates significantly, the update approximates $\Delta\theta_{t+1} \approx -\eta_t \hat{m}_t/\sigma_t$ which means the learning rate is proportional to the signal-to-noise ratio. Hence, the gradients scale with their mean in units of the standard deviation, which serves as a natural scale for the gradient update steps. In this thesis we make use of the ADAM optimizer (or a close variant) for all learning tasks.

In addition to optimizers that adjust the learning rate based on the gradient landscape, there are also many learning rate schedulers that change the learning rate η_t itself over time regardless of the gradient. Usually the learning rate change induced by a scheduler is much larger than the adjustment by the optimizer. Common learning rate schedulers include the decrease of the learning rate after a fixed amount of update steps or the exponential decay of the learning rate over time. For the training of the EPiC-CNF models in Chapter 7 we make use of the cosine annealing scheduler [167].



(a) Neuron **(b)** Neural network

Figure 3.1: Structure of a neural network. A single neuron **(a)** is composed of a linear transformation of the input features $\mathbf{x}$ and a non-linear activation function. Many neurons are connected in a neural network **(b)** which is composed out of multiple layers of neurons where the output of one layer is the input of the next. Figures taken from Reference [161].

3.2 Deep Neural Networks

Neural networks are non-linear models composed of many connected functions. The name is inspired by neurons connected in the human brain. The atomic unit of a neural network is a *neuroni* that takes an input a set of d features $\boldsymbol{x} = \{x_1, x_2, \ldots, x_d\}$ and computes a scalar output $a_i(\boldsymbol{x})$. A neural network is usually organized into *layers*, where each layer consists of a set of neurons that are computed in parallel. The layers are computed sequentially, i.e. the output of one layer is the input of the next layer. The first layer in a neural network is called the *input layer* and the final layer the *output layer*. The layers in between are called *hidden layers*. As every neuron is connected to every neuron in the adjacent layers, this type of neural network is called a *fully-connected network* (FCN) and the layers are often referred to as *fully-connected layers* or *dense layers*.

The function a_i can be decomposed into a linear function $z^{(i)}$ that weighs the input features and a non-linear *activation function* $\sigma_i(z^{(i)})$ that introduces non-linearity into the neural network. The linear transformation in a single neuron is usually a dot product with a weight vector $\boldsymbol{\omega}^{(i)} = \{\omega_1^{(i)}, \omega_2^{(i)}, \ldots, \omega_d^{(i)}\}$ and a bias $b^{(i)}$ addition. The transformation is given by

$$z^{(i)} = \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{x} + b^{(i)} = \mathbf{x}^T \cdot \mathbf{w}^{(i)} \tag{3.8}$$

where the bias term is expressed as another weight by adding a constant feature $x_0 = 1$ to the input vector $\mathbf{x} = (1, \boldsymbol{x})$ and defining $\mathbf{w}^{(i)} = (b^{(i)}, \boldsymbol{\omega}^{(i)})$. Including the activation function, the output of a single neuron is then given by the non-linear transformation

$$a_i(\mathbf{x}) = \sigma_i(z^{(i)}) = \sigma_i(\mathbf{x}^T \cdot \mathbf{w}^{(i)}). \tag{3.9}$$

The basic structure of a neural network is depicted in Figure 3.1.

In the past, common activation functions were step-functions (perceptrons), sigmoids, and hyperbolic tangents. Currently, more popular are rectified linear units (ReLUs) and variants like leaky rectified linear units (LeakyReLUs) and exponential linear units (ELUs). These

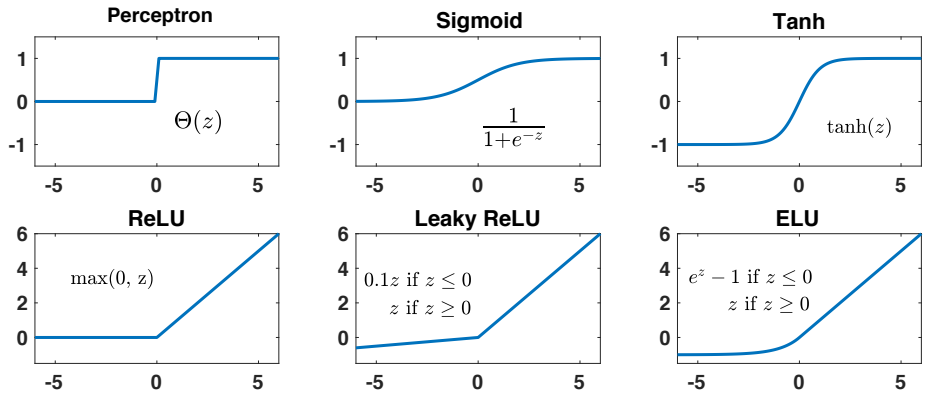


Figure 3.2: Several non-linear activation functions. Figures taken from Reference [161].

activation functions are depicted in Figure 3.2. When training a neural network with gradient descent-based optimizers, the various activation functions lead to different optimization properties. For sigmoid and hyperbolic tangent activation functions, the gradients are well-defined around the origin, but vanish for large inputs. This makes it difficult to train very deep networks. ReLU and its variants are often preferred, since for ReLU the gradients are well-defined for all positive inputs and for LeakyReLU the gradients are non-zero for negative inputs as well. This is why in this thesis for most models LeakyReLU is used. The sigmoid activation function is mostly used for the output of classification models (classifiers).

The general structure of a neural network is known as the network *architecture*. Figure 3.1b depicts the architecture of a simple *feed-forward neural network* composted of four layers: an input layer, two hidden layers, and an output layer. The overall network can be thought of as a complex non-linear function that maps inputs $\mathbf{x}$ to an output $\hat{y}$ depending on the learnable parameters $\mathbf{w}$. All trainable parameters are commonly called the *weights* of the network. The output of each network layer can be thought of as a feature representation of the input data and with increasing layers size and depth, the network can learn more complex representations of the data. In fact, it can be shown [168] that a feedforward neural network with a single hidden layer can approximate any continuous, multi-input/output function to an arbitrary accuracy given a sufficiently large number of neurons. However, empirically it is easier to train deeper and less wide networks than shallow and wide networks [169] and therefore modern neural networks usually contain many hidden layers. These are called *deep neural networks* (DNNs) which also motivates the term *deep learning*.

Particularly deep neural networks often contain so-called *skip connections* or *residual connections* which connect layers that are not directly adjacent to each other. These connections allow the network to access learned representations from earlier layers or to bypass layers completely. These residual connections are the core of the *ResNet* architecture [170] and the *U-Net* architecture [171] which are widely used in computer vision.

The choice of the network architecture and the functional form of its layers is highly dependent on the task and type of data that is used. For example, if the data contains certain symmetries, it is often advantageous to choose a layer type that is invariant under these symmetries to enforce that the network output is invariant and to make the optimization

task easier. Without an invariant network architecture, the network would need to learn these symmetries from the data which may reduce the model performance. For example, image data is often translation invariant which motivates the use of convolutional neural networks (see Section 3.4) and set data is permutation invariant which is exploited by a network architecture known as deep sets (see Section 3.5).

3.3 Training Deep Neural Networks

To train the weights $\mathbf{w}$ of a DNN model with common optimizers such as the ones discussed in Section 3.1, a loss function $L(\mathbf{w})$ needs to be specified that measures the model predictive performance appropriately. In case of supervised training, i.e. when the model is trained on labeled data, an example of the data set is given as $(\mathbf{x}_i, y_i)$ where y_i is the true property of $\mathbf{x}_i$ that the model should predict. The model prediction may be given as $\hat{y}_i(\mathbf{w})$. For regression problems, the loss function is often the mean squared error (MSE) also known as the L_2 norm:

$$L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i(\mathbf{w}))^2 \tag{3.10}$$

where n is the number of samples per minibatch.

For a classification problem, i.e. for categorical data, usually the cross-entropy loss is used as the model is usually a logistic classifier for binary data or a softmax classifier for multi-class data. For the binary case, the output of the DNN is the probability $\hat{y}_i(\mathbf{w}) = p(y_i = 1|\mathbf{x}_i; \mathbf{w})$ that the sample $\mathbf{x}_i$ belongs to class 1. The corresponding binary cross-entry loss for true labels $y_i \in \{0, 1\}$ and predicted probabilities is given by

$$L(\mathbf{w}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i(\mathbf{w})) + (1 - y_i)\log(1 - \hat{y}_i(\mathbf{w})). \tag{3.11}$$

When training the DNN, the loss function is minimized with respect to the model weights $\mathbf{w}$. To apply gradient-based optimizers, the gradients of the neural network need to be computed. This is done via a specialized algorithm known as *backpropagation* [172]. In general, backpropagation is simply the chain rule of calculus for practical differentiation and is outlined in the following [168].

3.3.1 Backpropagation

Considering a neural network that consists of M layers with the indices $m = 1, \ldots, M$, we denote the weight that connects the k-th neuron in layer $m-1$ to the j-th neuron in layer l as ω_{jk}^m and the bias of the latter neuron as b_j^m. The activation for this neuron can then be written as

$$a_j^m = \sigma\left(\sum_k \omega_{jk}^m a_k^{m-1} + b_j^m\right) = \sigma(z_j^m) \tag{3.12}$$

where σ is the activation function and z_j^m the linear transformation

$$z_j^m = \sum_k \omega_{jk}^m a_k^{m-1} + b_j^m. \tag{3.13}$$

The loss function L depends on the output activations a_j^M of the last layer M and therefore also on all the weights and biases in the previous layers. The error of the j-th neuron in the last layer M is expressed as

$$\Delta_j^M = \frac{\partial L}{\partial z_j^M} \tag{3.14}$$

and the error of a neuron in any layer m is given by

$$\Delta_j^m = \frac{\partial L}{\partial z_j^m} = \frac{\partial L}{\partial a_j^m}\sigma'(z_j^m) \tag{3.15}$$

where σ' is the derivative of the activation function with respect to the input evaluated at x. One can rewrite this error in terms of the partial derivative of the bias b_j^m as

$$\Delta_j^m = \frac{\partial L}{\partial z_j^m} = \frac{\partial L}{\partial b_j^m}\frac{\partial b_j^m}{\partial z_j^m} = \frac{\partial L}{\partial b_j^m} \tag{3.16}$$

since according to Equation 3.13 the partial derivative of the bias with respect to the linear transformation is $\partial b_m^m / \partial z_m^l = 1$. As the error Δ_j^m depends on the neurons in the subsequent layer $m+1$, the chain rule can be applied to rewrite the error as

$$\begin{aligned}\Delta_j^m = \frac{\partial L}{\partial z_j^m} &= \sum_k \frac{\partial L}{\partial z_k^{m+1}}\frac{\partial z_k^{m+1}}{\partial z_j^m} \\ &= \sum_k \Delta_k^{m+1}\frac{\partial z_k^{m+1}}{\partial z_j^m} = \left(\sum_k \Delta_k^{m+1}\omega_{kj}^{m+1}\right)\sigma'(z_j^m).\end{aligned} \tag{3.17}$$

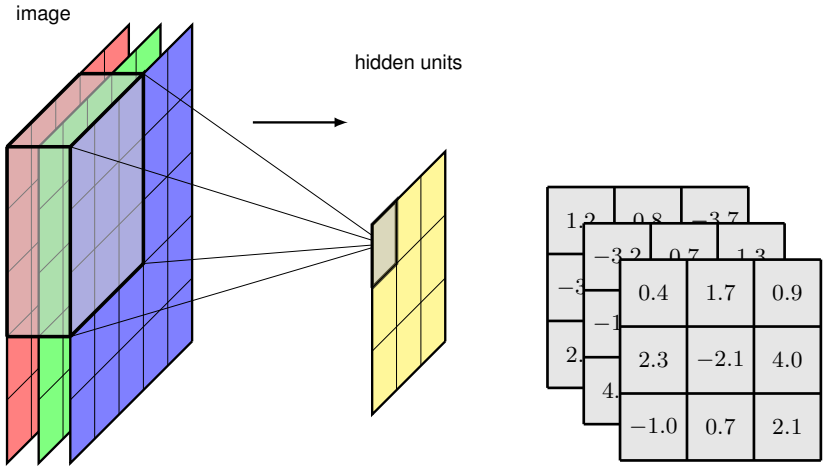
Finally, the partial derivative of the loss with respect to the weights ω_{jk}^m can be expressed as

$$\frac{\partial L}{\partial \omega_{jk}^m} = \frac{\partial L}{\partial z_j^m}\frac{\partial z_j^m}{\partial \omega_{jk}^m} = \Delta_j^m a_k^{m-1}. \tag{3.18}$$

Equipped with these equations, the overall backpropagation algorithm for calculation of all network gradients can be written in multiple steps as follows:

1. **Input Activations:** compute the activations a_j^1 for the input layer.
2. **Feedforward:** using Equations 3.12 and 3.13, compute the all a^m and z^m for all following layers.
3. **Output Error:** compute the error of the last layer M using Equation 3.15.
4. **Backpropagation:** using Equation 3.17, the error can be propagated backwards to calculate Δ_j^m for all layers.
5. **Gradient Calculation:** finally, calculte the gradients of the loss with respect to the weights $\partial L / \partial \omega_{jk}^m$ and biases $\partial L / \partial b_j^m$ using Equations 3.18 and 3.16.

With the backpropagation algorithm, the gradients of the loss functions with respect to all model parameters can be efficiently calculated as it only requires one "forward" pass of the model to calculate the output and the output error and then one "backward" pass



(a) Convolution operation **(b)** Kernel weights

Figure 3.3: **(a)** Illustration of the multidimensional convolutional filter that takes the input over the R, G, and B channels of an image. **(b)** Trainable kernel weights represented as a $3 \times 3 \times 3$ tensor. Figures taken from Reference [160].

to propagate the error back through the network for the gradient computations. Knowing the gradients, the model weights can be updated with an optimizer such as Adam. The backpropagation algorithm is essential for training deep neural networks and is implemented in all modern deep learning frameworks such as PyTorch [173], which was used to develop the models in this thesis.

3.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) [174] are a class of neural networks that are translation invariant and respect the locality of input data, which makes them particularly well-suited for predictive tasks on image data. An example for such a task could be the identification of objects in images regardless of their position in the image. A CNN is commonly made up of two types of layers: *convolutional layers* that apply a convolution operation to the input data with a set of learnable filters, and *pooling layers* that down-sample the input data while keeping the locality and spatial structure of the image. Usually, a CNN is composed by a sequence of convolutional layers and pooling layers before the intermediate output features are flattened and fed into a fully-connected neural network to compute the final output.

For 2D data like images, a layer l is defined by the height H_l, the width W_l, and the depth D_l, also known as the number of channels. The "area" (W_l, H_l) corresponds to the size of the 2D filter (or kernel) in the 2D plane and the number of channels C_l gives the number of filters in the layer. If the input is an image, the number of channels often corresponds to the three color channels R, G, and B. For 3D convolutional layers, there is an additional depth dimension D_l, hence the filter covers a "volume" (W_l, H_l, D_l). A square 2D filter of size F is defined by the three-dimensional trainable weight tensor of size $F \times F \times C_{l-1}$. The output of the convolutional layer is computed by sliding the tensor over the input data plane

and computing the scalar product of the tensor with the input data in its receptive field. A visualization of the convolution operation is shown in Figure 3.3. A hyperparameter called the *stride* S defines by how many neurons (pixels) the filter is translated between each operation, and it is common to pad the input with P zeros to keep the output size the same as the input size. With a large stride and little padding, the dimensionality of the image is reduced. After computing the filter output, a non-linearity such as a ReLU activation function is applied.

Using the scenario in Figure 3.3a as an example, a convolutional layer with a filter size of 3×3 and a stride of $S = 1$ applied to a 5×5 pixels image with 3 color channels results in an output of size of 5×5 with 1 color channel. We can pad the input with $P = 1$ zeros around the image resulting in a 7×7 input image and a 5×5 output image. We can further apply multiple filters such that the output has multiple channels. The output can be thought of as feature maps that represent certain properties of the input data. Many convolutional layers can be stacked to learn very complex features, i.e. a filter in an early layer could learn to detect edges or another filter in a later layer could learn to detect eyes in a picture of a face. Note, that the weights of the filters are *shared*, i.e. the same filter is applied to all locations of the input data. This property is what makes CNNs translation invariant and keeps the parameter count in check. The locality is preserved, since the filter output correlates with the input in its spatial vicinity. If the filters were not shared across the input, but initialized for each location, we speak of a *locally connected layer* which is not translation invariant and leads to a much higher number of weights.

In many CNNs, convolutional layers are intertwined with pooling layers that coarse-grain the spatial information by down-sampling the input. Typical pooling operations are *max pooling* or *average pooling* where either the maximum of the input or the average of the input over a small receptive field is taken. The pooling kernel for example could be of size 2×2 with a stride of 2 which would reduce an input image of size 32×32 to 16×16. Pooling layers are usually applied to reduce the dimensionality of the feature maps, which is especially important if their output is fed into a fully-connected network where parameter count of the first layer scale quadratically with the input image size. This kind of dimensionality reduction can also be achieved with convolutional layers and large strides with little padding, but pooling layers have the advantage of not introducing additional parameters to the model.

The reverse operation to a convolution layer is performed by a *transpose convolution* [175]. This layer can increase the dimensionality of the input and is often used in generative models to generate images from a low-dimensional noise vector. The trainable filters and their translation over the input image works similarly to a convolution layer, but the output is computed by summing every input pixel with every filter weight individually. The stride of the transpose convolution defines how much of the filter output is overlaid with adjacent filter outputs. A stride of 1 results in an output image of same dimensionality and a stride of larger than 1 yields an up-sampled image. For example, an original image with size 9×9 and a filter size of 3×3 with $S = 3$ would result in an output image of size 27×27. We use 3D convolutional layers in the encoder and 3D transpose convolutions in the decoder of the BIB-AE model in Chapter 5.

3.5 Graph- and Set-based Neural Networks

Neural networks that operate on graph-structured data are known as *graph neural networks* (GNNs). The simplest type of graph is simply a set of nodes (points) without any edges (pair-wise connections), this is known as a *set*. A set containing a discrete number of data points in some space is also often called a *point cloud*. The elements of a set are invariant under permutations, i.e. their order is undefined or irrelevant. If a predictive model based on a fully-connected neural network takes a set as input, it has to learn that the output should be invariant under permutations of the set entries which may hinder the optimization considerably. Therefore, it is advantageous to use a model with the *inductive bias* of permutation invariance. Generally, an inductive bias allows an algorithm to prioritize certain solutions over others, and in the context of neural networks, the inductive bias can be encoded into the network architecture itself. This is realized for example in the Deep Sets architecture [176]. Deep Sets and various implementations of GNNs can be generalized under the *graph network* (GN) framework as introduced in Reference [177].

In the GN framework, the computation is made up of multiple GN *blocks*, which takes a graph as an input and also outputs a graph. The *graph* G itself is defined as a 3-tuple

$$G = (\mathbf{u}, V, E) \tag{3.19}$$

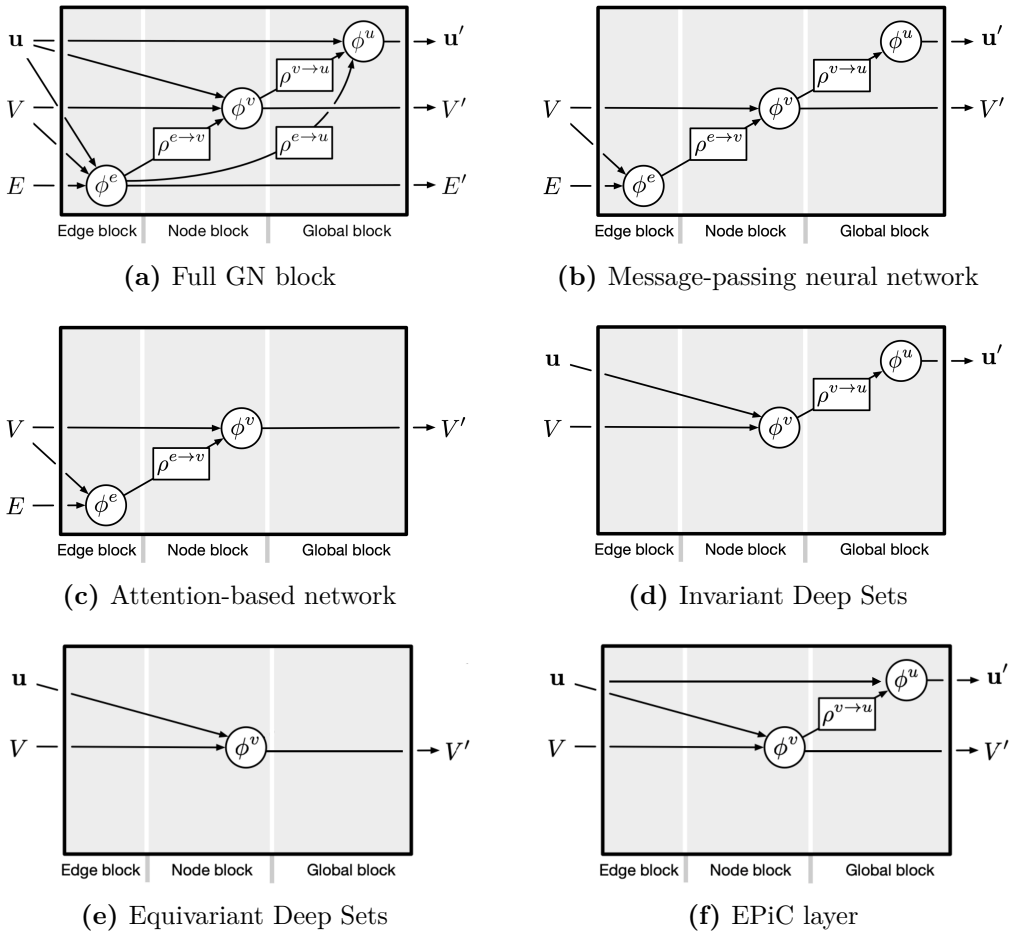
where $\mathbf{u}$ are global attributes of the graph, $V = \{\mathbf{v}_i\}_{i=1:N^v}$ is a set of nodes of cardinality N^v with attributes $\mathbf{v}_i$, and $E = \{\mathbf{e}_k, r_k, s_k\}_{k=1:N^e}$ is a set of edges of cardinality N^e with attributes $\mathbf{e}_k$, the index of the sender node s_k, and the index of the receiver node r_k. The GN block uses three functions to update the graph attributes, and three functions to aggregate node and edge features. These functions are given by

$$\begin{aligned} \mathbf{e}'_k &= \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) & \overline{\mathbf{e}}'_i &= \rho^{e\to v}\left(E'_i\right) \\ \mathbf{v}'_i &= \phi^v\left(\overline{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) & \overline{\mathbf{e}}' &= \rho^{e\to u}\left(E'\right) \\ \mathbf{u}' &= \phi^u\left(\overline{\mathbf{e}}', \overline{\mathbf{v}}', \mathbf{u}\right) & \overline{\mathbf{v}}' &= \rho^{v\to u}\left(V'\right) \end{aligned} \tag{3.20}$$

where $E'_i = \{\mathbf{e}'_k, r_k, s_k\}_{r_k=i,k=1:N^e}$ is the set of the updated edge attributes for each node i, $V' = \{\mathbf{v}'_i\}_{i=1:N^v}$ is the set of updated node attributes, and $E' = \{\mathbf{e}'_k, r_k, s_k\}_{k=1:N^e}$ is the set of all updated edge attributes. The functions ϕ^e, ϕ^v, and ϕ^u are the edge, node, and global update functions, respectively, and $\rho^{e\to v}$, $\rho^{e\to u}$, and $\rho^{v\to u}$ are the edge-to-node, edge-to-global, and node-to-global aggregation functions, respectively. The ϕ functions can be modeled by neural networks and the ρ functions are permutation invariant aggregations such as element-wise summation, averaging, or max-pooling. This way also the ϕ functions can be *invariant* under permutations of the respective node or edge features — here ϕ^v is invariant under edge permutation and ϕ^u is invariant under both edge and node permutation. Further, since the ϕ^v and ϕ^e functions are shared across all nodes and edges, their output is *equivariant* under node/edge permutation.

The computations in a GN block are visualized in Figure 3.4a and can be summarized with the following steps:

1. **Edge Update:** Apply ϕ^e to update the edge features based on the previous edge attributes, the sender and receiver node attributes, and the global attributes.



(a) Full GN block

(b) Message-passing neural network

(c) Attention-based network

(d) Invariant Deep Sets

(e) Equivariant Deep Sets

(f) EPiC layer

Figure 3.4: Different types of graph neural network configurations. Figures adapted from Reference [177].

2. **Edge-to-Node Aggregation:** Apply $\rho^{e\to v}$ to aggregate the updated edge attributes on a per-node basis.

3. **Node Update:** Apply ϕ^v to update the node attributes based on the previous node attributes, the aggregated edge attributes, and the global attributes.

4. **Node-to-Global Aggregation:** Apply $\rho^{v\to u}$ to aggregate the updated node attributes.

5. **Edge-to-Global Aggregation:** Apply $\rho^{v\to u}$ to aggregate all updated edge attributes.

6. **Global Update:** Apply ϕ^u to update the global attributes based on the previous global attributes, aggregated node attributes, and the aggregated edge attributes.

This is the sequence of computations as shown in the GN block depiction, but depending on the implementation, the order can be adapted.

A variety of graph- and set-based neural network architectures can be expressed as special cases of the GN framework. Several examples are depicted in Figure 3.4. The *message-passing neural network* (MPNN) [178] (Figure 3.4b) uses GN blocks without any global graph

attributes as input, no edge information is part of the output, and the global attribute is computed only based on an aggregation of node features. The eponymous "message function" is analogous to the edge update function ϕ^e but without any global information.

Attention-based neural networks can be expressed as the structure shown in Figure 3.4c. Here "attention" refers to the update of a node's features based on a weighted sum (of a function) of the attributes of the neighboring nodes. The edges, i.e. the weights between nodes, are calculated via a scalar pairwise function of the node attributes. Often, attention-based networks consider a fully-connected graph, i.e. all nodes are connected to all other nodes including themselves (self-attention). Often the in- and output of an attention block are only the node features and the edges are computed in every block anew, but the concept can be extended to incorporate general edge attributes as well [179]. A popular version of attention is *multi-headed self-attention* [180] where multiple ϕ^e and $\rho^{e\to v}$ functions – each with different neural network weights – take the same input and are computed in parallel before their output is concatenated. This way the different *heads* can focus on different aspects of the same input. Multi-headed self-attention constitutes the basis of the *transformer* architecture which is used in most modern image generation and language models.

The above-mentioned *Deep Sets* architecture is purely set-based and uses only global information and nodes attributes without any edges. Deep Sets comes in two variants: an invariant (Figure 3.4d) and an equivariant Deep Sets block (Figure 3.4e). In the invariant case, the model produces only a global output which could be for example a classification label. This way a permutation invariant classifier can be constructed. In high-energy physics, this network structure is used in the *particle flow networks* (PFNs) [181] for jet tagging. As jets are a set of particles, the Deep Sets paradigm is uniquely suited for this task. Additionally, since no edge features are calculated, the computational cost of such as classifier scales linearly with the set cardinality. In the equivariant Deep Sets case, the model produces a set of transformed node attributes instead of a global output. This output is optionally conditioned on a fixed global vector. This way the blocks can be easily stacked to learn more complex node representations and, for example, provide the input for an invariant Deep Sets model. The *ConcatSquash layer* (CSL) used in the PointWise Net of the CaloClouds models for calorimeter point cloud generation in Chapter 5 can be viewed as a realization of equivariant Deep Sets.

The *equivariant point cloud* (EPiC) [2] layer is closely related to Deep Sets and shown in Figure 3.4f. It performs an equivariant transformation of the node attributes and an invariant transformation of the global attributes. Together with the computational order and the specific pooling implementation, this makes the EPiC layer a novel contribution to the machine learning literature. It is used in the EPiC-GAN and EPiC-CNF models for jet generation in Chapter 7 and in the encoder of the CaloClouds model for calorimeter shower generation in Chapter 6. A detailed discussion of the EPiC layer is provided in Section 7.2.

Chapter 4

Generative Machine Learning

Many of the practical advancements in generative machine learning, also often referred to as *generative modeling*, have been developed in the field of computer vision. Therefore, some examples for the generative model implementations discussed in this chapter focus on image generation. However, these techniques are by no means limited to images and can be applied to any kind of data, including lists, time series, point clouds, or even (tokenized) text.

A generative model uses machine learning to learn a probability distribution for a set of training data. The trained model can then be used to generate new data samples from the learned distribution. In general, a generative model can be thought of as a distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$ where $\mathbf{x}$ is a data vector and $\boldsymbol{\theta}$ are learnable model parameters. For many applications, the generative model needs to be conditional, i.e. it is given by $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{c})$ where $\mathbf{c}$ is a condition vector. Using a particle physics example, $\mathbf{x}$ could be a particle jet and $\mathbf{c}$ the jet energy. For real-world data, such as photos or physics sensor data, the data vector space can be very complex. With the introduction of deep learning, it has become possible to train generative models on such high-dimensional data spaces with high fidelity. Therefore, these models are also often coined *Deep Generative Models* (DGMs).

Currently, the most common generative model families are generative adversarial networks (GANs), variational autoencoders (VAEs), normalizing flows, and diffusion models. Recently, consistency models have been introduced as a new generative modeling paradigm. In this thesis, we made use of all of these concepts, either as stand-alone models or in combination with each other.

As an overview, the following models are discussed in this thesis:

- The bounded information bottleneck autoencoder (BIB-AE) model in Chapter 5 uses the information bottlneck (IB) principle and combines a variational autoencoder (VAE), an adversarial autoencoder (AAE), and a Wasserstein generative adversarial network with gradient penalty (WGAN-GP).

- The CaloClouds model in Chapter 6 combines a VAE-like encoder, two normalizing flows, and a denoising diffusion probabilistic model (DDPM).

- The CaloClouds II model in Chapter 6 combines a normalizing flow and a diffusion model trained with score matching.

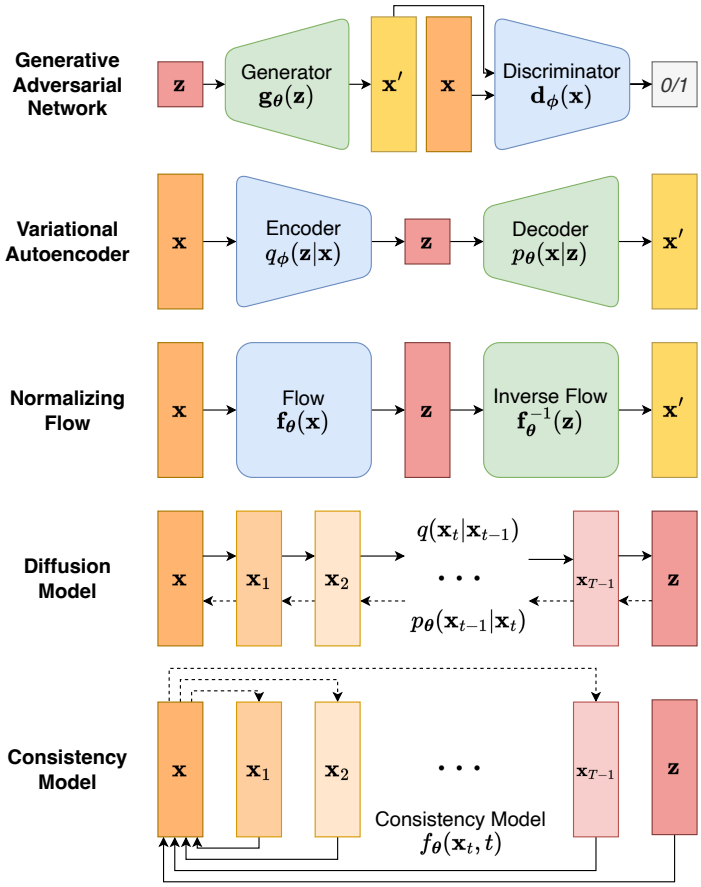


Figure 4.1: Overview of the different generative modeling paradigms used in this thesis.

- The CaloClouds II (CM) model in Chapter 6 combines a normalizing flow and a consistency model.
- The EPiC-GAN model in Chapter 7 is trained as a least squares GAN (LSGAN).
- The EPIC-CNF models in Chapter 7 are continuous normalizing flows (CNFs) trained with either the score matching or the flow matching objective.

An overview of these modeling paradigms is given in Figure 4.1. In the following sections all these models and some of their variants are outlined. A concise comparison of normalizing flows, variational autoencoders, and generative adversarial networks can be found in Reference [182]. A comprehensive overview of deep generative modeling is given in Reference [183]. This chapter follows largely the introduction to GANs, autoencoders, normalizing flows, and diffusion models outlined in Reference [160]. Generative modeling is a very fast-moving field with many new modeling paradigms and implementations being developed. Hence, some of the most state-of-the-art techniques like flow-matching and consistency models cannot be found yet in textbooks or literature reviews.

This chapter is structured as follows: Section 4.1 introduces generative adversarial networks (GANs) and two of their variants (LSGAN and WGAN) used in this thesis. The family

of autoencoders, including variational autoencoders (VAEs) and the bounded information bottleneck autoencoder (BIB-AE), is discussed in Section 4.2. Normalizing flows as well as continuous normalizing flows (CNFs) are introduced in Section 4.3. In Section 4.4, multiple parameterization of diffusion models and their relation to stochastic and ordinary differential equations are discussed. Finally, Section 4.5 introduces consistency models as the newest generative modeling paradigm used in this thesis.

4.1 Generative Adversarial Networks

A generative model can be considered as a non-linear transformation from a latent space $\mathbf{z}$ to a data space $\mathbf{x}$. The distribution of the latent space $p(\mathbf{z})$ may take the form of a Gaussian distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ and the non-linear transformation may be given as

$$\mathbf{x} = \mathbf{g}(\mathbf{z}, \boldsymbol{\theta}), \tag{4.1}$$

where $\mathbf{g}$ is a deep neural network with weights $\boldsymbol{\theta}$. In generative modeling, the neural network $\mathbf{g}$ is known as the *generator*. For brevity, we continue with writing the weights for neural networks in the style $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}) \equiv \mathbf{g}(\mathbf{z}, \boldsymbol{\theta})$.

The nonlinear transformation and the latent space together define a distribution over $\mathbf{x}$. For training, this model needs to be fitted to the training set $\{\mathbf{x}_n\}$ with $n = 1, \ldots, N$ and a total of N examples in the set. However, this fitting is complicated, because the likelihood function for optimizing $\boldsymbol{\theta}$ can generally not be evaluated in closed form. To the rescue comes the concept of *generative adversarial networks* (GANs) [184].

In a GAN, a second network, coined the *discriminator* $\mathbf{d}_{\boldsymbol{\phi}}$, is trained simultaneously with the generator $\mathbf{g}_{\boldsymbol{\theta}}$ and provides the gradients to update $\boldsymbol{\theta}$. While the goal of the generator is to generate synthetic ('fake') data indistinguishable from real data, the training objective of the discriminator is to distinguish the generated synthetic data from the real data. For the discriminator, this is achieved by training it with a simple classification loss function like binary cross-entropy. Thereby the loss of the generator is defined as maximizing the discriminator loss, i.e. by generating real-looking samples to make the classification task for the discriminator very difficult. One can therefore see the models $\mathbf{g}_{\boldsymbol{\theta}}$ and $\mathbf{d}_{\boldsymbol{\phi}}$ as *adversaries*. This training concept is visualized in Figure 4.2

To train the discriminator, a binary target t is defined with $t = 0$ for synthetic (fake) data and $t = 1$ for real data. The neural network $\mathbf{d}_{\boldsymbol{\phi}}$ with weights $\boldsymbol{\phi}$ uses as output activation a single logistic-sigmoid function, i.e. the output gives the probability of the data vector $\mathbf{x}$ is real as $P(t = 1) = \mathbf{d}_{\boldsymbol{\phi}}(\mathbf{x})$. The normalized cross-entropy loss function for training the discriminator is given by

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}) = -\frac{1}{N} \sum_{n=1}^{N} \left[t_n \log \mathbf{d}_n + (1 - t_n) \log(1 - \mathbf{d}_n) \right] \tag{4.2}$$

where $\mathbf{d}_n = \mathbf{d}_{\boldsymbol{\phi}}(\mathbf{x}_n)$ is the output of the discriminator for the data vector $\mathbf{x}_n$. The training set for the discriminator is a (usually balanced) mixture of real examples $\mathbf{x}_n$ and synthetic (fake) examples $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}_n)$ with $\mathbf{z}_n$ drawn from the latent distribution $p(\mathbf{z})$ (often a simple normal distribution).

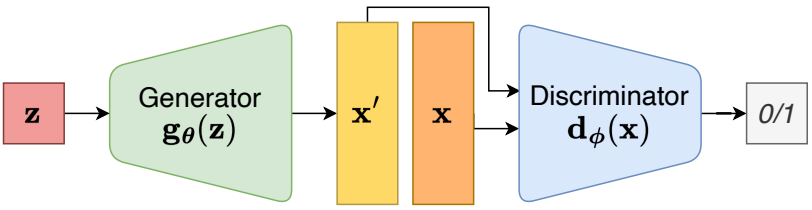


Figure 4.2: Illustration of a generative adversarial network (GAN). The generator $\mathbf{g}_{\boldsymbol{\theta}}$ generates synthetic data $\mathbf{x}'$ from a low-dimensional latent space $\mathbf{z}$. The generator is trained with an adversarial loss against the binary discriminator $\mathbf{d}_{\boldsymbol{\phi}}$ that tries to distinguish between real data $\mathbf{x}$ and the generated data $\mathbf{x}'$.

For the adversarial training to work, we train $\mathbf{d}_{\boldsymbol{\phi}}$ to maximize the probability of correctly distinguishing between real and fake samples. We simultaneously train $\mathbf{g}_{\boldsymbol{\theta}}$ to minimize the probability that $\mathbf{d}_{\boldsymbol{\phi}}$ can correctly identify the generated samples as fake, i.e. we minimize $\log(1 - \mathbf{d}_{\boldsymbol{\phi}}(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})))$. This initially seems a bit confusing, but it is the same statement as training the generator to maximize the probability that the discriminator falsely classifies the generated samples as real. The total GAN loss function can be written as

$$L_{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{real}}(\mathbf{x})} \left[\log \mathbf{d}_{\boldsymbol{\phi}}(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log(1 - \mathbf{d}_{\boldsymbol{\phi}}(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})))\right] \tag{4.3}$$

Here $\mathbb{E}$ denotes the expectation with its suffix indicating which variable is averaged over. For the adversarial training to work, the loss is maximized with respect to the discriminator weights $\boldsymbol{\phi}$ and minimized with respect to the generator weights $\boldsymbol{\theta}$.

In practice, the discriminator and generator are trained alternatingly each one or multiple steps before the other one is updated again. Loss convergence is usually not reached in GAN training since the generator and discriminator are ideally in a constant state of competition. A GAN "convergence" would imply a Nash equilibrium between the two, but this is rarely achieved in practice. Instead, the generator is regularly evaluated by a quantitative score or a visual inspection of the generated samples and the training is stopped once the desired quality is reached. When the GAN is done training, the discriminator is not used anymore and the generator can be used to generate synthetic data samples by sampling from the latent space. Further, a *conditional GAN* [185] can be implemented that sample from a conditional distribution $p(\mathbf{x}|\mathbf{c})$ where $\mathbf{c}$ is a condition vector. This way, the GAN can be conditioned to generate samples with specific properties such as a certain jet energy or particle type. The generator and discriminator networks then take the form $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{c})$ and $\mathbf{d}_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c})$, respectively.

While GANs are very effective in generating high-quality samples and can compete with more recent popularized generative approaches such as diffusion models (see Section 4.4) in image generation [186, 187], their training remains challenging. For instance, it is unclear when to best stop the training process. While one can track the training process with external measures, the generated sample quality often fluctuates strongly during training. Another often occurring challenge is *mode collapse*, where the generator only samples a small subset of valid outputs, i.e. always the same realistic image. If the discriminator does not learn to always reject those samples, the generator will continue to produce them and the training

process gets stuck. The training process can also be very difficult and slow if the generator is initialized such that it generates samples that are very different from the real data distribution. In this case, the discriminator can easily distinguish the samples but provides only very small gradients to the generator. To elevate these problems and to improve the overall training stability, several variants of the original GAN formulation have been developed.

4.1.1 GAN Variants

Many variants [188] of the training objective for GANs have been proposed. They differ in the loss functions used, but all keep the general concept of adversarial training. Two popular versions that are applied in this thesis are the *least squares GAN* (LSGAN) [189] and the *Wasserstein GAN* (WGAN) [190, 191].

LSGAN aims to stabilize the training by providing a stronger gradient. This is achieved by using a discriminator with a linear output activation and training it with a least-squares loss function instead of the cross-entropy loss. The general formulation of the LSGAN loss function is then given by

$$\begin{aligned} L_{\text{LSGAN}}(\boldsymbol{\phi}) &= \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{real}}(\mathbf{x})}\left[(\mathbf{d}_{\boldsymbol{\phi}}(\mathbf{x}) - b)^2\right] + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[(\mathbf{d}_{\boldsymbol{\phi}}(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})) - a)^2\right] \\ L_{\text{LSGAN}}(\boldsymbol{\theta}) &= \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[(\mathbf{d}_{\boldsymbol{\phi}}(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})) - c)^2\right] \end{aligned} \tag{4.4}$$

where a and b are the labels for the fake and real data, respectively, and c is the target value G wants D to output for the fake data. Opposed to a 'vanilla' GAN, both loss functions are minimized during the LSGAN training. In practice, it works well to adopt a 0-1 binary labeling scheme, i.e. $a = 0$ and $b = c = 1$. This scheme was used for the training of the EPiC-GAN in Chapter 7.

Another way of increasing the stability of a GAN is to use the distance between the generator distribution $p_G(x)$ and the real distribution $p_{\text{data}}(x)$ as a measure in the training. This is done in a WGAN by estimating the *Wasserstein-1 distance*, also known as the *earth mover's distance*, between the two distributions. The Wasserstein-1 distance arises from optimal transport theory and is defined as the minimum "cost" of transforming one distribution into the other. Since for high-dimensional data spaces, the Wasserstein distance is very difficult to compute, the WGAN [190] uses a discriminator $\mathbf{d}_{\boldsymbol{\phi}}$ with a linear output activation to approximate a surrogate. An improved version of the WGAN uses a *gradient penalty* (GP) term to stabilize the training process further. This model is known as *WGAN-GP* [191] and its loss function is given by

$$\begin{aligned} L_{\text{WGAN-GP}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = &- \mathbb{E}_{\mathbf{x}\sim p_{\text{real}}(\mathbf{x})}\left[\mathbf{d}_{\boldsymbol{\phi}}(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\mathbf{d}_{\boldsymbol{\phi}}(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}))\right] \\ &+ \lambda_{\text{GP}}\mathbb{E}_{\hat{\mathbf{x}}\sim p(\hat{\mathbf{x}})}\left[\left(\|\nabla_{\hat{\mathbf{x}}}\mathbf{d}_{\boldsymbol{\phi}}(\hat{\mathbf{x}})\|^2 - 1\right)^2\right] \end{aligned} \tag{4.5}$$

where λ_{GP} is a hyperparameter that weights the gradient penalty term and $\hat{\mathbf{x}}$ are random samples from the combined distribution of real and fake data $p(\hat{\mathbf{x}})$.

4.2 Autoencoder

A large topic in deep learning is representation learning, i.e. learning a representation of the data that is useful for subsequent applications. One well-known method for representation learning is the usage of *auto-associative neural networks*, also known as *autoencoders*. An autoencoder is a neural network that is trained to generate an output $\mathbf{y}$ that resembles closely its input $\mathbf{x}$. The internal representations the network learns, i.e. its latent space, can then be used to extract a representation $\mathbf{z}(\mathbf{x})$ of the data. Generally, an autoencoder network is split into two parts: the *encoder* network that maps the data to the latent space $\mathbf{z}(\mathbf{x})$, and the *decoder* network that maps the latent vectors back to the data space $\mathbf{y}(\mathbf{z})$. To avoid that the network simply learns to copy the input to the output, i.e. learns the identity function, a constraint on the network needs to be introduced. The most common constraint is to limit the dimensionality of the latent space, i.e. to introduce a *bottleneck* in between the encoder and decoder.

A simple linear autoencoder consisting just of two linear layers can at a global minimum achieve the same representation as a principal component analysis (PCA) with M components, where M is the dimensionality of the latent space. [192] Using deeper non-linear neural networks for the encoder and decoder, the autoencoder can learn more complex representations of the data, i.e. a non-linear form of PCA, but there is a risk that the optimization process gets stuck in a local minimum. The training of the autoencoder is done by minimizing a loss function that measures the difference between the input and output data. A common loss function is the mean squared error (MSE) or L2 loss:

$$L_{\text{AE}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\|\mathbf{y}_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{x}\|^2\right] \tag{4.6}$$

where $\mathbf{y}_{\boldsymbol{\theta}}$ is the autoencoder with weights $\boldsymbol{\theta}$ and $p_{\text{data}}(\mathbf{x})$ is the data distribution.

The autoencoder structure discussed so far is also sometimes known as a *deterministic autoencoder*. These are generally not used as generative models. In the following other kinds of autoencoders are introduced that can indeed be used for generative modeling.

4.2.1 Variational Autoencoder

The likelihood function for a latent-variable model is given by

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{4.7}$$

where the conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ over a D-dimensional data space is given by a deep neural network $\mathbf{d}_{\boldsymbol{\theta}}(\mathbf{z})$ and $p(\mathbf{z})$ is the prior distribution over an M-dimensional latent space. $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ will emerge as the generator or *decoder* of an autoencoder model and the prior can be given by a latent distribution over a normal Gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$. It is unfeasible to directly evaluate the likelihood Equation 4.7, but we can make use of an approximation to train a *variational autoencoder* (VAE) [193, 194], which can be used as a generative model. For training a VAE, three ingredients are needed: an evidence lower bound (ELBO) that is used to approximate the likelihood function, an encoder network that is used to approximate the posterior distribution that in turn can be used to calculate the ELBO

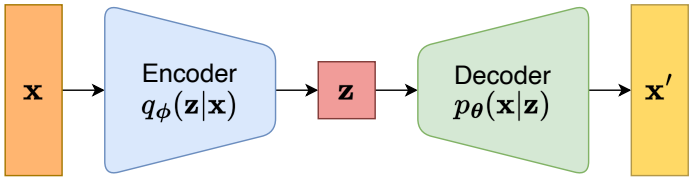


Figure 4.3: Illustration of a variational autoencoder (VAE). The encoder $p_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ maps the data $\mathbf{x}$ to the – usually lower-dimensional – latent space $\mathbf{z}$ and the decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ learns a reverse mapping.

and a parameterization known as the reparameterization trick to apply gradient descent to the ELBO. The general VAE structure is shown in Figure 4.3.

Evidence Lower Bound (ELBO)

To derive this likelihood approximation we consider another probability distribution $q_{\boldsymbol{\phi}}(\mathbf{z})$ over the latent space $\mathbf{z}$ with parameters $\boldsymbol{\phi}$. This distribution is also given by a neural network and will emerge as the encoder or inference network of the VAE. For any choice of $q_{\boldsymbol{\phi}}(\mathbf{z})$ the log-likelihood can be rewritten as [194]

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = L_{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \tag{4.8}$$

where L_{ELBO} is the *evidence lower bound (ELBO)* or *variational lower bound* and $D_{\text{KL}}(\cdot||\cdot)$ is the Kullback-Leibler divergence between two distributions. The ELBO is given by

$$L_{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}) \log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z})} \right) d\mathbf{z} \tag{4.9}$$

and the Kullback-Leibler divergence term is given by

$$D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) = -\int q_{\boldsymbol{\phi}}(\mathbf{z}) \log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z})} \right) d\mathbf{z} \tag{4.10}$$

where $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is the posterior distribution over the latent space given the data $\mathbf{x}$.

The Kullback-Leibler divergence D_{KL} measures the difference between two probability distributions, and it is always non-negative — although it is not symmetric and therefore not a true metric, i.e. it does not satisfy the triangle inequality. In general, with a probability distribution P and a probability distribution Q in the same sampling space $\mathcal{X}$ it is defined as:

$$D_{\text{KL}}(P||Q) = \int P(x) \log \left(\frac{P(x)}{Q(x)} \right) dx = -\int P(x) \log \left(\frac{Q(x)}{P(x)} \right) dx. \tag{4.11}$$

In Bayesian inference terms, the distribution Q is the prior distribution, and the distribution P is the posterior distribution. Hence, the D_{KL} measures the information loss when using the prior Q to approximate the posterior P. Technically in Equation 4.10 the reverse D_{KL} is used since it corresponds to $D_{\text{KL}}(Q||P)$.

Because the D_{KL} is always non-negative, the ELBO is a lower bound to the log-likelihood function, i.e. $L_{\text{ELBO}} \leq \log p_{\boldsymbol{\theta}}(\mathbf{x})$, and provides us with an approximation to the log-likelihood

function that can be evaluated with a Monte Carlo estimate and therefore used for model training.

For an independently and identically distributed (i.i.d.) training dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_n$ are samples from the model distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$, the log-likelihood is given by the sum of the ELBO and the D_{KL} terms:

$$\log p_{\boldsymbol{\theta}}(\mathcal{D}) = \sum_{n=1}^{N} L_{\mathrm{ELBO},n}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \sum_{n=1}^{N} D_{\mathrm{KL}}(q_{\boldsymbol{\phi},n}(\mathbf{z}_n)||p_{\boldsymbol{\theta},n}(\mathbf{z}_n|\mathbf{x}_n)) \tag{4.12}$$

where

$$L_{\mathrm{ELBO},n}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \int q_{\boldsymbol{\phi}}(\mathbf{z}_n) \log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n)}{q_{\boldsymbol{\phi}}(\mathbf{z}_n)} \right) d\mathbf{z}_n. \tag{4.13}$$

This indicates that there are separate latent variables $\mathbf{z}_n$ for each data vector $\mathbf{x}_n$ and therefore each latent variable is associated with an independent distribution $q_{\boldsymbol{\phi},n}(\mathbf{z}_n)$. We will exploit this in Chapter 5 to analyze the distributions of information-rich latent variables.

Hence, maximizing the log-likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ is equivalent to maximizing the ELBO L_{ELBO} or to minimizing the D_{KL} term, i.e. to minimize the difference between $q_{\boldsymbol{\phi}}(\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

Encoder or Amortized Inference

The latter implies that the log-likelihood is maximized when $q_{\boldsymbol{\phi}}(\mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, i.e. the Kullback-Leibler divergence is $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) = 0$. This is achieved by optimizing $q_{\boldsymbol{\phi}}(\mathbf{z})$ to approximate the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. For this to work, we implement *amortized inference* which requires that the model q needs to be conditioned on the data $\mathbf{x}$, so we replace $q_{\boldsymbol{\phi}}(\mathbf{z})$ with $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. This network is called the *encoder* or *inference model* as it maps the input data space to the latent space.

Usually, the encoder is chosen such that it produces a Gaussian distribution with a diagonal covariance matrix where the mean μ_j and the variance σ_j^2 are given by the output of the neural network:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \prod_{j=1}^{M} \mathcal{N}(\mathbf{z}_j|\mu_j(\mathbf{x}, \boldsymbol{\phi}), \sigma_j^2(\mathbf{x}, \boldsymbol{\phi})). \tag{4.14}$$

Now we have described both models in the ELBO and can maximize it with respect to the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ to train the VAE.

Reparameterization Trick

One problem remains: the ELBO Equation 4.13 is intractable as we cannot properly back-propagate through the sampling process of the latent variables $\mathbf{z}_n$. Equation 4.13 can be rewritten as

$$\begin{aligned} L_{\mathrm{ELBO},n}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \int q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n) \log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n)}{q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n)} \right) d\mathbf{z}_n \\ &= \int q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n) \log p_{\boldsymbol{\theta}}(\mathbf{x}_n|\mathbf{z}_n) d\mathbf{z}_n - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n)||p(\mathbf{z}_n)) \end{aligned} \tag{4.15}$$

where the first term can be interpreted as the reconstruction error and the second term as a regularizer. Since the second term is the D_{KL} between two Gaussian distributions, it can be calculated analytically as [193]

$$D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n)||p(\mathbf{z}_n)) = -\frac{1}{2}\sum_{j=1}^{M}\left(1+\log(\sigma_j^2(\mathbf{x}_n)) - \mu_j^2(\mathbf{x}_n) - \sigma_j^2(\mathbf{x}_n)\right). \tag{4.16}$$

In principle, this approach works with any latent prior distribution that is analytically tractable, but we consider here only the simple case of a Gaussian distribution. The first term can be approximated with a Monte Carlo estimate:

$$\int q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n)\log p_{\boldsymbol{\theta}}(\mathbf{x}_n|\mathbf{z}_n)d\mathbf{z}_n \simeq \frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\mathbf{x}_n|\mathbf{z}_n^{(l)}) \tag{4.17}$$

where $\mathbf{z}_n^{(l)}$ are sampled from the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n)$ and $l = 1,\ldots,L$ are the sample indices. By drawing samples from the encoder, we can optimize the decoder, but we cannot yet optimize the encoder.

To resolve this issue, the so-called *reparameterization trick* is applied: Considering the random variable ϵ drawn from a normal Gaussian, the quantity $z = \mu + \sigma\epsilon$ is also normally distributed with mean μ and variance σ^2. This parameterization is applied to the outputs of the encoder $\mu_j(\mathbf{x}_n)$ and $\sigma_j^2(\mathbf{x}_n)$. Instead of drawing samples from the encoder, we draw ϵ from a standard normal distribution and then transform it with the encoder outputs:

$$z_{nj}^{(l)} = \mu_j(\mathbf{x}_n) + \sigma_j^2(\mathbf{x}_n)\epsilon_n^{(l)}j \tag{4.18}$$

This allows the gradients with respect to $\boldsymbol{\phi}$ to be calculated and for backpropagation through both the decoder and encoder networks.

The full loss function that shall be minimized during the VAE training, i.e. the upper bound to the negative log-likelihood, can finally be written as

$$L_{\mathrm{VAE}}(\boldsymbol{\theta},\boldsymbol{\phi}) = -\sum_n\left[\frac{1}{2}\sum_{j=1}^{M}\left(1+\log(\sigma_{nj}^2) - \mu_{nj}^2 - \sigma_{nj}^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\mathbf{x}_n|\mathbf{z}_n^{(l)})\right] \tag{4.19}$$

where $\mathbf{z}_n^{(l)}$ has elements $z_{nj}^{(l)} = \mu_{nj} + \sigma_{nj}\epsilon^{(l)}$ with $\mu_n j = \mu_j(\mathbf{x}_n)$ and $\sigma_{nj} = \sigma_j(\mathbf{x}_n)$ as the output of the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n)$, and n elements in the mini-batch. In practice, the log-likelihood term for the reconstruction error is often replaced with a reconstruction error suitable for the specific task, such as a mean squared error like in the regular autoencoder loss Equation 4.6. After the model is trained the encoder may be discarded and the decoder can be used as a generative model by sampling the latent space from the prior distribution $p(\mathbf{z})$ and using the neural network to map it back to the data space.

The first term of the loss function Equation 4.19 moves the encoder distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ towards the Gaussian prior distribution $p(\mathbf{z})$. This enables the decoder to produce realistic-looking samples when sampling directly from the prior distribution instead of the encoded space. However, when during training the encoder converges towards the prior, i.e. $D_{\mathrm{KL}} = 0$, the latent space carries zero information and the decoder cannot use it anymore to reconstruct the data. This is known as *posterior collapse* and leads to a bad reconstruction performance.

On the other hand, when the D_{KL} term is too high, the latent space cannot be properly approximated by the prior distribution during sampling. This leads to a very good reconstruction performance of the VAE, but when sampling novel data from the prior, the generated samples do not resemble the data distribution.

Both of these problems can be tackled by weighting the D_{KL} term and the reconstruction term appropriately. This is done by introducing a hyperparameter β that weights the terms in the loss function. This concept is known as the *β-VAE* [195]. In Chapter 5 we will study the impact of the posterior collapse on the generation quality of BIB-AE (a VAE-like model discussed below) and examine an alternative to the prior sampling. Another method of preventing posterior collapse is to clip the D_{KL} term to a certain minimum value. This method is introduced for the encoder of the CaloClouds model in Chapter 6.

4.2.2 Adversarial Autoencoder

The *Adversarial Autoencoder* (AAE) [196] was proposed as an alternative way to the VAE for training an autoencoder as a generative model. The model combines ideas from an autoencoder with a GAN, i.e. it is trained using a reconstruction error as well as an adversarial loss to regularize the latent space. Just like a VAE, the AAE consists of an encoder and a decoder network, but it includes also an additional discriminator network. This discriminator is used to distinguish between the latent space distribution and an arbitrary prior distribution. The encoder learns to map the data space to this prior latent distribution and tries to fool the discriminator. The decoder is used to map the latent space back to data and after training can be used as a generative model by sampling from the prior distribution.

Let $p(\mathbf{z})$ be the prior distribution of the latent space and $p_{\mathrm{data}}(\mathbf{x})$ the data distribution. Often the prior is defined by a normal Gaussian distribution like in out example of the VAE, but the advantage of an AAE is, that it can be any distribution, even if it is not analytically tractable. We further define three distributions that are given by deep neural networks: the encoder $\mathbf{e}_{\boldsymbol{\phi}}(\mathbf{x})$ with weights $\boldsymbol{\phi}$ that maps the data space to the latent space, the decoder $\mathbf{d}_{\boldsymbol{\theta}}(\mathbf{z})$ with weights $\boldsymbol{\theta}$ that maps the latent space back to the data space, and the binary discriminator $\mathbf{c}_{\boldsymbol{\omega}}(\mathbf{z})$ with weights $\boldsymbol{\omega}$ that distinguishes between the prior distribution and the encoder distribution.

All three models are simultaneously trained, i.e. the encoder and decoder are trained as an autoencoder with a reconstruction loss, and the encoder and discriminator are trained with an adversarial loss. The weight updates occur in multiple stages per mini-batch: first, the encoder and decoder are updated with the reconstruction loss between data and reconstructed samples, then the discriminator is updated with the cross-entropy loss between samples from the prior and encoded samples, and finally, the encoder is updated to confuse the discriminator.

The optimization of the encoder $\mathbf{e}_{\boldsymbol{\phi}}(\mathbf{x})$ and decoder $\mathbf{d}_{\boldsymbol{\theta}}(\mathbf{z})$ is done by minimizing the reconstruction loss such as an MSE (analogous to Equation 4.6):

$$L_{\mathrm{AE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}(\mathbf{x})}\left[(\mathbf{d}_{\boldsymbol{\theta}}(\mathbf{e}_{\boldsymbol{\phi}}(\mathbf{x})) - \mathbf{x})^2\right]. \tag{4.20}$$

Optimization of the discriminator $\mathbf{c}_{\boldsymbol{\omega}}(\mathbf{z})$ and the encoder $\mathbf{e}_{\boldsymbol{\phi}}(\mathbf{x})$ (as additional objective) is

done analogous to the GAN loss Equation 4.3:

$$L_{\text{adv}}(\boldsymbol{\omega}, \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log \mathbf{c}_{\boldsymbol{\omega}}(\mathbf{z})\right] + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\log(1 - \mathbf{c}_{\boldsymbol{\omega}}(\mathbf{e}_{\boldsymbol{\phi}}(\mathbf{x})))\right] \tag{4.21}$$

where the function is maximized with respect to the discriminator weights $\boldsymbol{\omega}$ and minimized with respect to the encoder weights $\boldsymbol{\phi}$.

An advantage of the AAE is the arbitrary choice of prior distribution, which may be chosen to fit well for a certain data space. In Reference [196] a generalization of the reparameterization trick is introduced that allows for the backpropagation through the sampling process for non-Gaussian priors. A disadvantage of the AAE might be unstable training due to the adversarial loss. This problem may be mitigated by using alternative GAN implementations such as WGAN or LSGAN.

4.2.3 Bounded Information Bottleneck Autoencoder (BIB-AE)

The information bottleneck (IB) method [197] is an information-theoretical principle introduced for determining how much *relevant information* a random variable $X \sim p_x$ contains about another statistically dependent random variable Y. Considering the joint distribution of these variables $p(X, Y)$, the relevant information is defined by the *mutual information* $I(X;Y)$. The mutual information can be given in terms of the Kullback-Leibler divergence via

$$I(X;Y) = D_{\text{KL}}(p(X,Y)||p(X) \otimes p(Y)) \tag{4.22}$$

where $p(X) \otimes p(Y)$ is the product distribution of the marginals $p(X)$ and $p(Y)$, i.e. the joint distribution if the variables were independent. An optimal representation of X would capture all relevant aspects that are necessary to predict Y while compressing (discarding) the irrelevant features. The relevant information of X shall be denoted by $\hat{X}$. Finding the optimal representation of $\hat{X}$ is done by minimizing the Lagrangian

$$\mathcal{L}[p(\hat{x}|x)] = I(X;\hat{X}) - \beta I(\hat{X};Y) \tag{4.23}$$

where the Lagrangian multiplier β scales the trade-off between the complexity of the representation $I(X;\hat{X})$ and the amount of preserved relevant information $I(\hat{X};Y)$. Here the Lagrangian is used in the context of Lagrangian multipliers, i.e. functions that shall be optimized under certain constraints. Hence, the Lagrangian can be viewed as a loss function for neural network optimization.

The IB principle can be used for the interpretation of deep neural networks from an information-theoretical perspective [198]. It can also be used to gain a nuanced understanding of certain types of generative models. In Reference [199] the IB method is used to study various VAE and GAN implementations. This results in the introduction of an overarching framework termed the *bounded information bottleneck autoencoder (BIB-AE)*, that encompasses the VAEs and GANs as special cases. Generative models based on the VAE and GAN principles can be viewed through the IB principle as all introducing different bounds on mutual information terms as regularizers. The main difference in the models can therefore be viewed by the choice of target objective, i.e. how to judge the fidelity of the model, and the choice of the regularizer, i.e. what modes of information compression restrict the model.

We define a generative model (a decoder in the autoencoder setup) as $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ which generates a data distribution $p_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim p_{\boldsymbol{\theta}}(\mathbf{z})}\left[p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]$ where $p_{\boldsymbol{\theta}}(\mathbf{z})$ is a target distribution of the latent space. An inference model (an encoder in the autoencoder setup) is defined as $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ with $q_{\boldsymbol{\phi}}(\mathbf{z})$ as the prior "true" distribution of the latent space. The empirical data distribution is given by $p_{\text{data}}(\mathbf{x})$.

Analogous to the Lagrangian of the IB principle, we can define for unsupervised models Lagrangian of the BIB-AE as

$$\mathcal{L}_{\text{BIB-AE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = I_{\boldsymbol{\phi}}(X; Z) - \beta I_{\boldsymbol{\phi},\boldsymbol{\theta}}(Z; X) \tag{4.24}$$

where X is a data vector and Z is a latent vector. The mutual information depends on the model parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

The Lagrangian can be rewritten as [199]

$$\begin{aligned}\mathcal{L}_{\text{BIB}-\text{AE}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = &\underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}(\mathbf{x})}}\left[D_{\text{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z})\right)\right]}_{\text{A}} - \underbrace{D_{\text{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z})\|p_{\boldsymbol{\theta}}(\mathbf{z})\right)}_{\text{B}} \\ &-\beta[\underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}(\mathbf{x})}}\left[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left(p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right)\right]\right]}_{\text{C}} - \underbrace{D_{\text{KL}}\left(p_{\text{data}}(\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{x})\right)}_{\text{D}}].\end{aligned} \tag{4.25}$$

Here, the reconstruction terms (C) and (D) ensure the fidelity of the model and the terms (A) and (B) regularize the latent space. The BIB-AE parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are optimized by minimizing the Lagrangian.

In the context of GANs and VAEs, the terms are implemented in the following ways:

(A) The term denotes the D_{KL} between the encoder and the target latent distribution. It is commonly implemented in a VAE to regularize the latent space and can be analytically solved for Gaussian distributions.

(B) The term denotes the D_{KL} between the prior latent distribution and the target latent distribution. It is implemented in an AAE with a latent space discriminator or with a maximum mean discrepancy (MMD) [200] term.

(C) Describes the log-likelihood of the decoder that reconstructs the encoded latent space. It corresponds to the reconstruction error in an autoencoder and may be implemented for example with an MSE.

(D) The term denotes the D_{KL} between the data distribution and the generated distribution. This term is implemented in a GAN by the discriminator network that distinguishes between real and synthetic samples.

A detailed analysis of the utilization of one or multiple of these terms in generative models such as VAE, β-VAE, AAE, InfoVAE [201], GAN, and VAE-GAN [202] is given in Reference [199].

We used the BIB-AE principle to design a novel generative model for calorimeter shower simulations [6] that uses multiple model accuracy measures such as MSE and adversarial training and multiple regularizers for optimal information compression such as the D_{KL} and adversarial training to constrain the latent space. The model is introduced in Chapter 5 and subsequently used for an analysis of the information compression of calorimeter showers in its latent space.

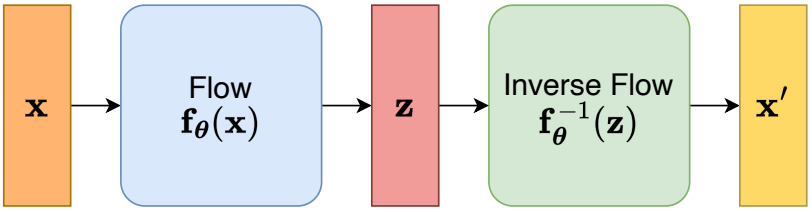


Figure 4.4: Illustration of a normalizing flow as a generative model, where the flow $\mathbf{f}_{\boldsymbol{\theta}}$ maps data to a Gaussian latent space and the inverse flow $\mathbf{f}_{\boldsymbol{\theta}}^{-1}$ can be used to generate new data samples.

4.3 Normalizing Flows

Another way to train a non-linear transformation from a latent space to a data space is by using *normalizing flows* (NFs) [203–205]. The goal for these types of models is to restrict the form of the transformation (the neural network) such that the likelihood function can be evaluated directly. For this purpose, we define a latent distribution $p_{\mathbf{z}}(\mathbf{z})$ — also known as the *base distribution* for flows — with latent vectors $\mathbf{z}$ together with the function $\mathbf{x} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z})$ that maps the latent space to the data space, where $\mathbf{f}_{\boldsymbol{\theta}}$ is a neural network with weights $\boldsymbol{\theta}$. Usually the base distribution is chosen to be very simple, i.e. a normal Gaussian distribution, such that novel latent space samples $\mathbf{z}^* \sim p_{\mathbf{z}}(\mathbf{z})$ can be easily sampled to generate new data samples $\mathbf{x}^* = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}^*)$. Hence, $f_{\boldsymbol{\theta}}$ can be viewed as a generator network.

For the calculation of the likelihood function, a data space distribution is needed that depends on the inverse of the neural network function $f_{\boldsymbol{\theta}}$. The inverse function shall be given by $\mathbf{z} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}))$. This definition requires that the $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\theta}}$ are bijective functions, i.e. they are invertible. A special kind of neural network architecture is needed to allow such invertibility. These are discussed below. With this formulation, each value of the data space $\mathbf{x}$ has a unique value in the latent space $\mathbf{z}$ and vice versa. This also implies that the data space and the latent space have the same dimensionality. The general idea for using normalizing flows as generative models is illustrate in Figure 4.4.

The change of variables for multivariant distributions can be used to determine the data density:

$$p_{\mathbf{x}}(\mathbf{x}|\boldsymbol{\theta}) = p_{\mathbf{z}}(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{x})) \left|\det \mathbf{J}(\mathbf{x})\right| \tag{4.26}$$

where $\mathbf{J}(\mathbf{x})$ is the Jacobian matrix whose elements are given by the partial derivatives

$$J_{ij}(\mathbf{x}) = \frac{\partial g_{\boldsymbol{\theta},i}(\mathbf{x})}{\partial x_j}. \tag{4.27}$$

Since the dimensionality of $\mathbf{x}$ and $\mathbf{z}$ are equal, the models can be very large and computationally expensive for high-dimensional data such as images. Further, the evaluation of $\det \mathbf{J}(\mathbf{x})$ generally scales with the dimensionality D as $\mathcal{O}(D^3)$, which can be very costly. Therefore, in addition to invertibility further restrictions on the model are imposed to calculate the determinant efficiently.

For training the neural network, we need to maximize the log-likelihood or equivalently minimize the negative log-likelihood. For a training set of $\mathcal{D} = \mathbf{x}_1, \ldots, \mathbf{x}_N$ data points, we

can use Equation 4.26 to write the log-likelihood function as

$$\begin{aligned}\log p(\mathcal{D}|\boldsymbol{\theta}) &= \sum_{n=1}^{N} \log p_{\mathbf{x}}(\mathbf{x}_n|\boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \left[\log p_{\mathbf{z}}(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{x}_n)) + \log\left|\det \mathbf{J}(\mathbf{x}_n)\right|\right].\end{aligned} \tag{4.28}$$

To ensure that the neural network $f_{\boldsymbol{\theta}}$ is invertible, every layer of the network needs to be invertible by itself since for several layers of successive transformations

$$\mathbf{x} = \mathbf{f}^A(\mathbf{f}^B(\mathbf{f}^C(\mathbf{z}))) \tag{4.29}$$

their inverse is given by

$$\mathbf{z} = \mathbf{g}^C(\mathbf{g}^B(\mathbf{g}^A(\mathbf{x}))) \tag{4.30}$$

where $\mathbf{g}^{i\in\{A,B,C\}}$ are the inverse transformations of $\mathbf{f}^{i\in\{A,B,C\}}$. This also allows for an easy evaluation of the Jacobian determinants for each layer by using the chain rule:

$$J_{ij} = \frac{\partial g_i}{\partial x_j} = \sum_k \sum l \frac{\partial g_i^C}{\partial g_k^B} \frac{\partial g_k^B}{\partial g_l^A} \frac{\partial g_l^A}{\partial x_j}. \tag{4.31}$$

Hence, the log-determinant of the Jacobian matrix is given by the sum of log-determinants for each layer.

This type of modeling is known as normalizing flows because it allows transforming a complex data distribution into a "normalized" form such as a simple Gaussian latent distribution, and it uses a sequence of invertible mappings that lets the transformation "flow" in between the data and latent spaces.

4.3.1 Coupling Flows

To implement a normalizing flow, we need to define invertible neural network layers which in sequence can be used to create the flow transformation. A popular flow architecture that is used in this thesis is a *coupling flow* [203]. A coupling flow such as the RealNVP model [206] is made up of *coupling layers* and *permutation layers*.

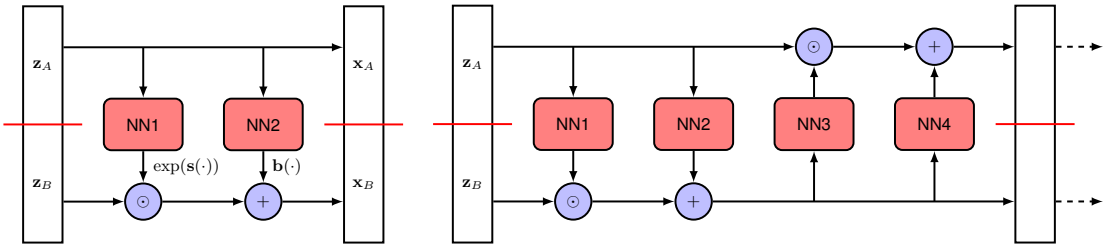
In the following, an example of such a coupling layer is given. Consider the input $\mathbf{z}$ and output $\mathbf{x}$ of a layer to be partitioned into two parts, i.e. $\mathbf{z} = [\mathbf{z}_A, \mathbf{z}_B] = [\mathbf{x}_{1:d}, \mathbf{x}_{d+1:D}]$ where $\mathbf{z}$ has dimensionality D and $\mathbf{z}_A$ dimensionality d (the same split is performed for $\mathbf{x}$). The output of a coupling layer is given in two parts. One is simply a copy of the input:

$$\mathbf{x}_A = \mathbf{z}_A \tag{4.32}$$

and the second part is a linear transformation:

$$\mathbf{x}_B = \exp(\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{z}_A)) \odot \mathbf{z}_B + \mathbf{b}_{\boldsymbol{\phi}}(\mathbf{z}_A) \tag{4.33}$$

where $\mathbf{s}_{\boldsymbol{\phi}}$ and $\mathbf{b}_{\boldsymbol{\phi}}$ are non-linear transformations, i.e. neural networks, with weights $\boldsymbol{\phi}$, and $\odot$ denotes the Hadamard product, i.e. element-wise multiplication. $\mathbf{s}$ and $\mathbf{b}$ can be implemented



(a) Single layer **(b)** Two layers permuted

Figure 4.5: Example of an invertible coupling architecture for normalizing flows. **(a)** shows a single coupling layer and **(b)** two coupling layers with the second one permuted. Shown is the forward direction, but the transformation is easily invertible. Figures taken from Reference [160].

as two separate networks with different weights or as two outputs of the same network. This formulation is easily invertible as

$$\mathbf{z}_A = \mathbf{x}_A \tag{4.34}$$

$$\mathbf{z}_B = \exp(-\mathbf{s}_\phi(\mathbf{z}_A)) \odot (\mathbf{x}_B - \mathbf{b}_\phi(\mathbf{z}_A)). \tag{4.35}$$

The forward direction of this transformation is depicted in Figure 4.5a.

This allows also a simplification in calculating the Jacobian determinant by splitting the Jacobian matrix into four parts:

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_d & \mathbf{0} \\ \frac{\partial \mathbf{z}_B}{\partial \mathbf{x}_A} & \operatorname{diag}(\exp(-\mathbf{s})) \end{bmatrix}. \tag{4.36}$$

The Jacobian matrix is now a lower triangular matrix and as such its determinant is simply given by the product of the diagonal elements, i.e. $\det \mathbf{J} = \prod_i \exp(-s_i)$.

To increase the flexibility of such a flow, multiple coupling layers are stacked. However, the value of $\mathbf{x}_A = \mathbf{z}_A$ would not be changed. Therefore, *permutation layers* are introduced in between the single coupling layers. For two coupling layers with a reverse permutation in between, this is illustrated in Figure 4.5b. With this architecture, it is possible to build complex normalizing flows that can be trained with the negative log-likelihood loss.

In a generalized form, Equation 4.33 can be written as

$$\mathbf{x}_B = \mathbf{h}(\mathbf{z}_B, g_\theta(\mathbf{z}_A)) \tag{4.37}$$

where $\mathbf{h}(\cdot)$ is an efficiently invertible function, known as the *coupling function*, and $g_\theta(\mathbf{z}_A)$ is a neural network, known as the *conditioner*.

4.3.2 Continuous Normalizing Flows

Another approach to normalizing flows is to apply "infinitely" deep neural networks that are defined in terms of ordinary differential equations (ODEs). These are called *neural ODEs* [207] and can be used to construct a *continuous normalizing flow* (CNF).

The idea is to define a neural network with a very large amount of layers approaching infinity. A single layer of said network may be defined with a residual connection such as

$$\mathbf{z}^{(t+1)} = \mathbf{z}^{(t)} + \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}^{(t)}) \tag{4.38}$$

where $t = 1, \ldots, T$ are the labels of the sequential layers. To keep the parameter count finite, the weights $\boldsymbol{\theta}$ of the neural network are shared across all layers. When exploring the limit of infinite layers, the change introduced by a single layer because infinitesimally small. This allows us to define the vector $\mathbf{z}$ as a function of continuous variable t, i.e. $\mathbf{z}(t)$, with its evolution described by a differential equation

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t)) \tag{4.39}$$

where the variable t is usually called the "time". This differential equation is known as a *neural ordinary differential equation* [207] ('ordinary' because it depends on a single variable).

The input of the network may be given by $\mathbf{z}(t = 0)$, then the output at time $t = T$ is obtained by integrating the neural ODE:

$$\mathbf{z}(T) = \mathbf{z}(0) \int_0^T \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t))dt. \tag{4.40}$$

This integral can be solved by using any numerical integration method, such as the Euler method given by Equation 4.38. In such a 'continuous network' the number of integration steps (ODE evaluation steps) can be seen as the depth of the network and generally the predictions improve with more steps. Non-linear ODE solvers and with fixed or adaptive step sizes can be applied to approximate the ODE with fewer steps and network evaluations.

With the neural ODE Equation 4.39 we can implement a *continuous normalizing flow (CNF)* by defining a base distribution over the input vector $p(\mathbf{z}(0))$ and then transforming it to the target distribution $p(\mathbf{z}(T))$ by integrating the neural ODE with a standard ODE solver. It can be shown [207] that such a transformation can be evaluated by integrating a differential equation

$$\frac{d \log p(\mathbf{z}(t))}{dt} = -\mathrm{Tr}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{z}(t)}\right) \tag{4.41}$$

where $\frac{\partial \mathbf{f}}{\partial \mathbf{z}}$ is the Jacobian matrix with elements $\frac{\partial f_i}{\partial z_j}$ and $\mathrm{Tr}(\cdot)$ denotes the trace of the matrix. This non-linear transformation is called a CNF and is depicted in Figure 4.6. CNFs can be trained with the *adjoint sensitivity method* [208] that are essentially a continuous version of backpropagation. Just like in a regular normalizing flow, the CNF for density estimation is trained with the maximum likelihood estimation analogous to Equation 4.28. The training, however, is computationally expensive and memory intensive as it requires evaluating an ODE solver with a sufficient number of steps for every weight update.

Evaluating the Jacobian determinant in a (discrete) normalizing flows model scales with $\mathcal{O}(D)$ when it is implemented with layers such as coupling blocks. Evaluating a CNF requires calculating the trace of the Jacobian matrix ($\mathcal{O}(D)$), but as every element of the matrix requires solving the ODE which scales with $\mathcal{O}(D)$ itself, the overall scaling behavior of a CNF is generally $\mathcal{O}(D^2)$. The scaling can be brought down to $\mathcal{O}(D)$ by approximating the trace

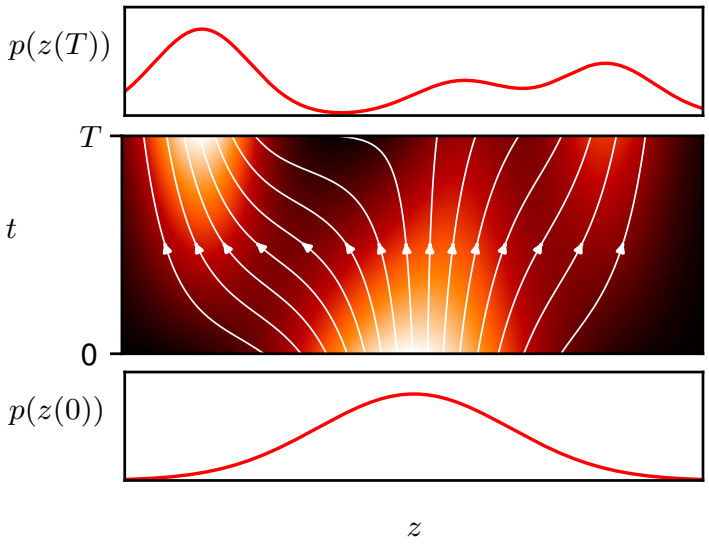


Figure 4.6: Illustration of a continuous normalizing flow (CNF). The transformation of the Gaussian base distribution at time $t = 0$ to the target distribution at time $t = T$ is shown. The density of continuous flow lines indicates the density of the target distribution. Figure taken from Reference [160].

with a Monte Carlo estimator known as *Hutchinson's trace estimator* which for a matrix $\mathbf{A}$ is given by

$$\mathrm{Tr}(\mathbf{A}) = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\boldsymbol{\epsilon}^T\mathbf{A}\boldsymbol{\epsilon}\right] \tag{4.42}$$

where $\boldsymbol{\epsilon}$ is a random vector sampled from a normal distribution. For M samples this results in a trace approximation of

$$\mathrm{Tr}(\mathbf{A}) \simeq \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{\epsilon}_m^T\mathbf{A}\boldsymbol{\epsilon}_m. \tag{4.43}$$

With $M = 1$, this results in a noisy Jacobian trace estimate, but it can be still sufficient as part of a stochastic gradient descent optimization. This estimator was introduced for the implementation of an efficient CNF called FFJORD [209].

Recently, the *flow matching* method [210–212] was introduced as a more efficient way for training CNFs. With flow matching there is no need for backpropagation with an integrator which reduces the memory footprint further and allows for more stable training while also keeping the sampling via ODE solvers fast and flexible. The method is closely related to score matching and diffusion models and is discussed in Section 4.4.3.

4.4 Diffusion Models

For many applications in generative modeling (i.e. text-to-speech models, image generation) *diffusion models* [213, 214], also known as *score-based generative models* [215], have emerged as the current state-of-the-art, outperforming GANs in generative fidelity [216] usually at the cost of increased computational complexity. The general idea behind diffusion models is the corruption of an input data sample by adding noise to it sequentially over a large number of

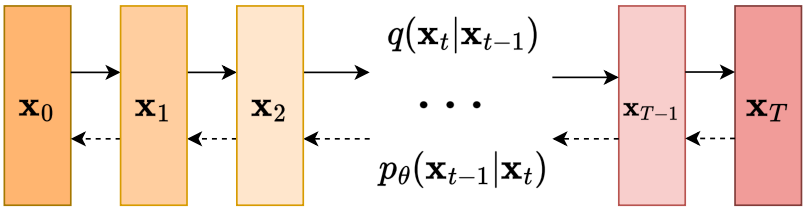


Figure 4.7: Illustration of the general diffusion model framework with the forward diffusion process $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ and the reverse denoising process $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

steps until the data sample is indistinguishable from random noise. This corruption process can be seen analogously to an encoding process in an autoencoder setup, although in diffusion models this encoder is a fixed Gaussian distribution. A deep neural network is trained to learn to revert this process, i.e. a decoder is trained that learns to remove a bit of noise from the data sample at each step until the data sample is reconstructed, or a novel sample is generated. The diffusion framework provides stable training for generative models and can generate high-fidelity samples. As the generation process potentially involves thousands of denoising steps, it might however be computationally expensive. A graphical illustration of a diffusion model is shown in Figure 4.7.

There are many variants of diffusion models with more being developed as this is currently a very active field of research in computer vision and machine learning. The approach was popularized by the *denoising diffusion probabilistic model* [214] which implements a diffusion model that predicts the noise that needs to be removed from an input sample (instead of the data sample itself) and is trained on a fixed number of denoising steps (usually $\mathcal{O}(1000)$). The model has strong connections to denoising *score matching* [215, 217] which describes the diffusion process with a stochastic differential equation (SDE). This allows for solving the denoising process with standard numerical differential equation solvers and therefore with a variable number of model evaluations for faster sample generation.

A process to streamline the training of diffusion models and relate them to CNFs was introduced with *flow matching* [210–212]. The flow matching objective allows for a very stable training and fast data sampling with standard ODE solvers. To mitigate the computational cost of sampling from diffusion models, various *distillation* methods [218] are explored to create variants of the models that are more efficient to evaluate without compromising the generative fidelity. To this end, a computationally particular efficient method is the distillation of a score-based generative model into a consistency model [219] which allows single and multi-step generation. All these methods are introduced in this section and are utilized in the following chapters for the generation of calorimeter showers and particle jets.

4.4.1 Denoising Diffusion Probabilistic Models

To introduce the first type of diffusion model, termed denoising diffusion probabilistic model (DDPM), we consider the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and latent variables $\mathbf{x}_1, \ldots, \mathbf{x}_T$ with the same dimensionality. The index denotes the "time" t just like in the CNF case and T is the total number of diffusion steps, i.e. the total number of latent vectors. It is common in diffusion models to use the index zero in $\mathbf{x}_0$ to denote the data space and the index T in $\mathbf{x}_T$ to denote the latent variable with the maximum noise, i.e. where the corrupted data is indistinguishable from random noise. Note that the latent spaces previously were denoted with $\mathbf{z}$, but to be consistent with the common diffusion model literature, we use $\mathbf{x}_T$ here. The following derivation follows Reference [214].

The DDPM model takes the form of a latent variable model $p_{\boldsymbol{\theta}}(\mathbf{x}_0) = \int p_{\boldsymbol{\theta}}(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$ with the trainable parameters $\boldsymbol{\theta}$. Here $\mathbf{x}_{0:T}$ is shorthand for $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$. The *reverse process*, i.e. the denoising process "noise to data", is characterized by the joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_{0:T})$ with a Markov chain of learned Gaussian transition as

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{0:T}) = p_{\boldsymbol{\theta}}(\mathbf{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t). \tag{4.44}$$

The chain starts with Gaussian noise $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$ and each link in the Markov chain is given by

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)) \tag{4.45}$$

where the Gaussian transitions $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ are the outputs of a neural network (the "decoder") which take the current latent variable $\mathbf{x}_t$ and the current time step t as input with $1 < t \leq T$.

The *forward process* or *diffusion process*, i.e. the encoding process "data to noise", is characterized by the approximate posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ which is a Markov chain made up of a fixed Gaussian process

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{4.46}$$

with the Gaussian transitions according to the variance schedule $\beta_1, \ldots, \beta_T$:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}). \tag{4.47}$$

The variances β_t are typically set hyperparameters according to a specific scheduler function. An important property for training is that the forward process allows for sampling any $\mathbf{x}_t$ directly without the need to sample all previous steps:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I}) \tag{4.48}$$

where $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ with $\alpha_t = 1 - \beta_t$. Overall the forward process has no trainable parameters and only the reverse process is learned. In the DDPM implementation [214], the variance of the reverse process is also fixed depending on the scheduled variance β_t as $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \sigma^2 \mathbf{I}$ where σ^2 is function of β_t. So the generative neural network is only used to approximate the mean of the Gaussian noise.

The DDPM model is trained in a similar way as the VAE by maximizing the ELBO since the log-likelihood cannot be directly evaluated. The ELBO to the log-likelihood is given by

$$\mathbb{E}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]. \tag{4.49}$$

As a loss function, the negative log-likelihood is minimized and the ELBO can be rewritten as the upper bound leading to the general DDPM loss function:

$$L_{\text{DDPM}}(\boldsymbol{\theta}) = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \tag{4.50}$$

with the first term being simply the negative log-likelihood of the initially sampled Gaussian noise to start the denoising process. Note that since the q encoding distribution is fixed, there is no need for the reparameterization trick like in a VAE setup.

The ELBO can be rewritten in terms of the KL divergence and since both the forward and reverse processes are written in terms of Gaussians, the KL divergence can be calculated in closed form like in the VAE case. For a specific time step, the loss can be evaluated with a mean squared error (MSE) as

$$L_{\text{DDPM},t-1}(\boldsymbol{\theta}) = \mathbb{E}_q\left[\frac{1}{2\sigma^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|^2\right] + \text{const.} \tag{4.51}$$

where the constant second term does not depend on $\boldsymbol{\theta}$ and the forward process posterior $\boldsymbol{\mu}_t$ is given by a linear combination of the data sample and the sample at the current time step as

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\overline{\alpha}_t}\beta_t}{1-\overline{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\overline{\alpha}_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{x}_t. \tag{4.52}$$

Hence, the neural network $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ is trained to predict the mean of the forward process posterior, i.e. the Gaussian noise that needs to be removed from the data sample at each time step. Empirically, the loss can further be simplified to [214] the final DDPM training loss:

$$L_{\text{DDPM,simple}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right] \tag{4.53}$$

where t is uniformly sampled between 1 and T, $\boldsymbol{\epsilon}$ is sampled from a standard Gaussian distribution, and $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is a neural network that predicts $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$ (where $\mathbf{x}_t$ is here the corrupted data sample as $\mathbf{x}_t = \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}$). During training, the t is sampled individually for every data sample in the mini-batch which means that multiple time steps are optimized for in parallel. Note that the neural network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ needs to have the same input and output dimensionality as the data (not counting the conditioning t), therefore often a *U-Net* architecture [171] is used.

For generation, first $\mathbf{x}_T$ is sampled, then the reverse process is sequentially applied as

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z} \tag{4.54}$$

where $\mathbf{z}$ is sampled from a standard Gaussian distribution. This process is successively applied from $t = T, \ldots, 1$ until a novel data sample is generated. Only in the final step to $\mathbf{x}_0$ the second term $\sigma_t\mathbf{z}$ is omitted to generate a noise-free data sample.

While the DDPM model can create high-fidelity data samples, the sampling is computationally expensive considering the number of model evaluations needed. An alternative approach to diffusion is the convert the reverse process into a differential equation that can be solved with adaptive for fixed-step numerical solvers. This approach is discussed in the next section.

4.4.2 Score Matching

Generative models trained via *score matching* [217, 220] are closely related to diffusion models, but introduce them from a different perspective. This is why diffusion models are also often called *score-based generative models*. Both DDPM and the score-based model introduced in Reference [217] are based on a discrete denoising process with potentially thousands of steps. This naturally leads to the question of what would happen in the infinite limit of steps, i.e. a continuous denoising process. It turns out, that one can describe the denoising process with a stochastic differential equation (SDE) [215] which allows for a much more flexible description of diffusion models. In the following, first the concept of score-based generative modeling and score matching is introduced. Then the connection to continuous denoising processes with SDEs is discussed.

The *score function*, *Stein score*, or simply *score* is defined as the gradient of the log-likelihood of a probability distribution $p(\mathbf{x})$ with respect to the data vector $\mathbf{x}$, i.e. it is given by the vector-valued function

$$\mathbf{s}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}). \tag{4.55}$$

Since the score function has the same dimensionality as the data, for images one can visualize the score as a vector field pointing towards the direction of the highest local data density. Knowing the score function means one is able to model the original data density and this means one can use it to generate novel data samples. In a score-based generative model, this is done by approximating the score with a deep neural network $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$ with trainable parameters $\boldsymbol{\theta}$. Since the score model needs to have the same dimensionality as the data, often a U-Net architecture is used.

To train the score model, one can simply minimize the distance between the real data score and the model with a simple MSE loss such that the loss function becomes

$$L_{\mathrm{SM}}(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}(\mathbf{x})}\left[\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\mathrm{data}}(\mathbf{x})\|^2\right]. \tag{4.56}$$

However, since the real score function is not known, this loss cannot be directly optimized. This is where methods for score matching [220] comes into play. It can be shown that the above objective is equivalent to

$$L_{\mathrm{SM}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}(\mathbf{x})}\left[\mathrm{tr}(\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) + \frac{1}{2}\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\|^2)\right] \tag{4.57}$$

where the Jacobian of the score model is given by $\nabla_{\mathbf{x}}\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$. Since for high-dimensional data the trace of the Jacobian is expensive to compute, a popular approximation is the application of *denoising score matching* [221]: A pre-defined noise distribution $q_\sigma(\hat{\mathbf{x}}|\mathbf{x})$ is used to corrupt the data sample $\mathbf{x}$ such that score matching can be applied to estimate the score of the

perturbed data distribution $q_\sigma(\hat{\mathbf{x}}) = \int q_\sigma(\hat{\mathbf{x}}|\mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}$. This leads to the denoising score matching loss

$$L_{\text{DSM}}(\boldsymbol{\theta}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}),\hat{\mathbf{x}}\sim q_\sigma(\hat{\mathbf{x}}|\mathbf{x})}\left[\|\mathbf{s}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}) - \nabla_{\hat{\mathbf{x}}} \log q_\sigma(\hat{\mathbf{x}}|\mathbf{x})\|^2\right]. \tag{4.58}$$

However, the approximation $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ is only valid for small noise levels such that $q_\sigma(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$.

A common choice for the noise distribution is a Gaussian distribution $q_\sigma(\hat{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\hat{\mathbf{x}}|\mathbf{x}, \sigma^2\mathbf{I})$. With this Gaussian perturbation, one can rewrite the score function as

$$\nabla_{\hat{\mathbf{x}}} \log q_\sigma(\hat{\mathbf{x}}|\mathbf{x}) = -\frac{1}{\sigma^2}\boldsymbol{\epsilon} \tag{4.59}$$

where $\boldsymbol{\epsilon} = \hat{\mathbf{x}} - \mathbf{x}$ is sampled from a normal distribution. Considering the noise model for the forward process of DDPM in Equation 4.48, this is equivalent to

$$\nabla_{\hat{\mathbf{x}}} \log q_\sigma(\hat{\mathbf{x}}|\mathbf{x}) = -\frac{1}{1-\overline{\alpha}_t}\boldsymbol{\epsilon}. \tag{4.60}$$

With this formulation, it can be seen that the loss function in Equation 4.58 is the MSE between the score model output and the noise vector $\boldsymbol{\epsilon}$. Hence, the denoising score matching loss has the same form as the DDPM loss in Equation 4.53 and the score model $\mathbf{s}_{\boldsymbol{\theta}}$ plays the same role as the noise predictor model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ in DDPM up to a given scaling factor $-1/(1-\overline{\alpha}_t)$.

To implement the training based on the denoising score matching loss, one needs to choose the variance σ^2 for the noise that smears out the data distribution. However, a too large variance might corrupt the data too much and leads to a poor approximation of the score function. Therefore, in Reference [217] it proposed to use a sequence of noise values $\sigma_1 < \sigma_2 < \cdots < \sigma_T$ where σ_1 is small enough to not distort the data distribution much and σ_T is large enough to mitigate problems in the score estimation such as too low data density regions. This σ_i sequence can be seen as a schedule for the noise level and can be implemented with the β_t schedule in DDPM.

For an accurate score estimation, the score model then takes σ as an additional input similar to the time step t conditioning in DDPM. The score network is hence given by $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$ which is known as a *noise conditional score network* (NCSN) [217]. Finally, the loss function for training the NCSN via denoising score matching can be written as

$$L_{\text{DSM}}(\boldsymbol{\theta}, \sigma) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma) + \frac{1}{\sigma^2}\boldsymbol{\epsilon}\|^2\right]. \tag{4.61}$$

and combined over all $\sigma \in \{\sigma_i\}_{i=1}^T$ yields

$$L_{\text{DSM}}(\boldsymbol{\theta}, \{\sigma_i\}_{i=1}^T) = \frac{1}{T}\sum_{i=1}^{T} \lambda\sigma_i L_{\text{DSM}}(\boldsymbol{\theta}, \sigma_i) \tag{4.62}$$

where $\lambda(\sigma_i)$ are weighting coefficients. Hence, this training procedure mirrors closely the DDPM training objective.

Once trained, new samples can be generated with the score model by sampling from the model sequentially with $i = T, T-1, \ldots, 2, 1$. This sampling procedure from the score can be described in terms of *Langevian dynamics* and is therefore known as *annealing Langevian dynamics*. Overall the sampling process is analogous to the generating with the DDPM model.

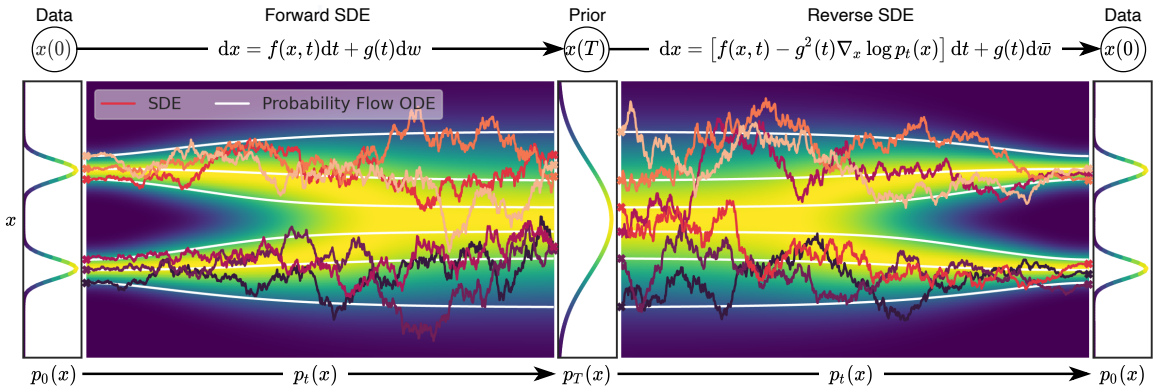


Figure 4.8: Illustration of the forward and reverse diffusion process of a score-based generative model described by a stochastic differential equation (SDE). Additionally, the associated deterministic probability flow ODE trajectories are shown. Figure taken from Reference [215].

Stochastic Differential Equations

In general, the more steps are used in the denoising process, the higher fidelity data samples can be generated. Therefore, it is natural to ask a similar question as was discussed in the context of discrete normalizing flows: What happens in the limit of an infinite number of steps? It turns out, that similar to flows where the continuous limit is described by a neural ODE, the continuous limit of denoising steps is described by a stochastic differential equation (SDE) [215]. Both the DDPM and the score-based model can be seen as discretizations of the continuous SDE model.

The forward diffusion process can be modeled with a general SDE of the form

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w} \tag{4.63}$$

where $t \in [0, T]$ is a continuous time variable, $\mathbf{f}$ is a deterministic vector valued function called the *drift* coefficient, $g(t)$ is a scalar function called the *diffusion* coefficient, and $\mathbf{w}$ is the standard Wiener process also known as Brownian motion. While the drift term is deterministic (like in an ODE), the diffusion term is stochastic and given by infinitesimal Gaussian steps.

The corresponding reverse diffusion process is also given by an SDE

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\tilde{\mathbf{w}} \tag{4.64}$$

where $\tilde{\mathbf{w}}$ is the Wiener process in reverse time from T to 0 and $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is the score function at time t. The forward and reverse diffusion processes in terms of the SDEs are illustrated in Figure 4.8.

The score function can be estimated with a neural network and trained with score matching as discussed above. The continuous version of the denoising score matching loss is given by

$$L_{\text{SDE}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_t|\mathbf{x}_0}\left[\lambda_t \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_{0t}(\mathbf{x}_t|\mathbf{x}_0)\|^2\right] \tag{4.65}$$

where λ_t is a time-dependent weighting function and t is uniformly sampled from $[0, T]$, $\mathbf{x}_0$ is sample from the training data distribution, and $p_{t0}(\mathbf{x}_t|\mathbf{x}_0)$ is the forward transition kernel from time step $t = 0$ to t (usually a fixed Gaussian process that is easily evaluated).

A trained score model can be used to generated new samples by solving Equation 4.64 in reverse time. This can be done with numerical solvers that discretize the time variable. Among the simplest solvers is the *Euler-Maruyama* solver [222] which uses a pre-determined number of equally spaced time steps. This results in a sampling procedure that mirrors the DDPM sampling and the score-based modeling via Langevian dynamics. The advantage of the reverse SDE formulation is that it allows to apply more complex SDE solvers that can improve the overall generative fidelity with fewer model evaluations.

Any diffuison process expressed as an SDE has also an associated deterministic process expressed as an ODE. This ODE is called the *probability flow ODE* and its trajectories have the same marginal probability densities $\{p_t(\mathbf{x})\}_{t=0}^{T}$ as the SDE. The probability flow ODE of the reverse process corresponds to Equation 4.64 without the stochastic Wiener process and is given by

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt. \tag{4.66}$$

When the score in this ODE is approximated by a neural network, it is an example of a neural ODE. Form this perspective, a continuous normalizing flow can be implemented using a score model. For sample generation, the advantage of the ODE formulation is that it allows for the application of very efficient (adaptive and fixed step) numerical ODE solvers that can potentially generated high-fidelity samples with significantly fewer model evaluations than the SDE solvers. It also allows for an exact log-likelihood evaluation as possible with normalizing flows.

Expressing diffusion models very generally through an SDE allows for a flexible study of different model implementation choices. Such as study was performed in Reference [223] where the authors did a systematic scan over various ODE and SDE samplers, the associated noise levels of the SDE solvers, and different noise parameterization, schedulers, and weighting functions. From this scan, suggestions for efficient sampler and training parameterization are derived which we adapted for the CaloClouds II model used in Chapter 6.

4.4.3 Flow Matching

A diffusion model with a neural network that models the score function can be sampled from with numerical ODE solvers. This effectively results in a continuous normalizing flow (CNF). However, the training still is based on a stochastic diffusion process with the score matching objective. Flow matching [210–212] was introduced as an alternative training method for CNFs based on deterministic probability paths. It allows for a very stable and fast training of diffusion models resulting in models than can be very efficiently sampled from with standard ODE solvers. In the following, flow matching is introduced as well as its per-sample training objective *conditional flow matching* [212] — analogous to score matching that is implementable with the denoising score matching objective.

To introduce flow matching , we write the neural ODE Equation 4.39 in terms of a time-dependent vector field $\mathbf{v}(t)$ that is used to construct a time-dependent mapping $\boldsymbol{\phi}_t(\mathbf{x})$

which is the *flow*. The flow is a time-dependent function of the data that maps the data $\mathbf{x}$ to a specific time step, i.e. $\mathbf{x}_t = \boldsymbol{\phi}_t(\mathbf{x})$. The neural ODE is then given by

$$\frac{d\boldsymbol{\phi}_t(\mathbf{x})}{dt} = \mathbf{v}_t(\boldsymbol{\phi}_t(\mathbf{x})). \tag{4.67}$$

As introduced in Section 4.3.2, the vector field $\mathbf{v}_t$ is usually given by a neural network with trainable parameters $\boldsymbol{\theta}$, and we write it in accordance with the previous notation as $\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}, t)$. The neural network $\mathbf{v}_{\boldsymbol{\theta}}$ becomes then a model of the flow $\boldsymbol{\phi}_t$ and is known as the CNF.

Unlike Reference [212], we continue to use the time step $t = 0$ to denote the data space and $t = T$ to denote the maximally corrupted space, i.e. the Gaussian latent space. In the following we set the maximum time step $T = 1$ to make the linear interpolation between $t = 0$ and $t = 1$ more intuitive.

In Reference [207] the training of $\mathbf{v}_{\boldsymbol{\theta}}$ is done with the adjoint sensitivity method which is computationally expensive since it requires many ODE evaluations. A novel way to train the CNF is inspired by score matching: We consider a target probability density path $p_t(\mathbf{x})$ with a corresponding vector field $\mathbf{u}_t(\mathbf{x})$ that allows to map p_1 to p_0, where p_1 is a simple distribution such as a standard Gaussian and p_0 is the data distribution $p_{\text{data}}(\mathbf{x})$. The flow matching objective is now simply to minimize the distance between the model vector field $\mathbf{v}_{\boldsymbol{\theta}}$ and the target vector field $\mathbf{u}_t$ via an MSE loss:

$$L_{\text{FM}}(\boldsymbol{\theta}) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\mathbf{x}\sim p_t(\mathbf{x})}\left[\|\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}, t) - \mathbf{u}_t(\mathbf{x})\|^2\right]. \tag{4.68}$$

However, a priori, the appropriate choices for the target vector field $\mathbf{u}_t$ and $p_t(\mathbf{x})$ are not known.

In Reference [212] it is shown that an equivalent objective can be derived which is called *conditional flow matching* (CFM), where the condition are examples from the data space, i.e. the training set. The CFM objective is given by

$$L_{\text{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\mathbf{x}_0\sim p_{\text{data}}(\mathbf{x}_0),\mathbf{x}\sim p_t(\mathbf{x}|\mathbf{x}_0)}\left[\|\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}, t) - \mathbf{u}_t(\mathbf{x}|\mathbf{x}_0)\|^2\right] \tag{4.69}$$

where $p_t(\mathbf{x}|\mathbf{x}_0)$ is a conditional probability path starts in data space at $t = 0$ and ends in Gaussian space at $t = 1$ with the associated vector field $\mathbf{u}_t(\mathbf{x}|\mathbf{x}_0)$. Interestingly, the CFM objective works with any choice of conditional probability path and vector field.

We see once again the strong relation of this objective to diffusion models, where the conditional probability trajectory is given by the forward diffusion process. The final CFM objective bears close resemblance to the DDPM loss in Equation 4.53 and the denoising score matching loss in Equation 4.61 with the difference that here the model is not trained to predict the original sampled noise, but the perturbed data at each time step. This approach to training diffusion models is also implemented in various other diffusion model parameterization and among the suggestions in Reference [223].

Among other possibilities, a straightforward choice for the trajectory is introduced in Reference [212] with *optimal transport conditional vector fields*. Considering Gaussian conditional probability paths of the form

$$p_t(\mathbf{x}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_t(\mathbf{x}_0), \sigma_t(\mathbf{x}_0)^2\mathbf{I}) \tag{4.70}$$

where $\boldsymbol{\mu}$ is the time-dependent mean and σ is the time-dependent scalar standard deviation of a Gaussian distribution, the associated conditional flow can be expressed as

$$\boldsymbol{\psi}_t(\mathbf{x}) = \sigma_t(\mathbf{x}_0)\mathbf{x} + \boldsymbol{\mu}_t(\mathbf{x}_0). \tag{4.71}$$

To ensure that all conditional paths converge to the same Gaussian distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I})$ at $t = 1$, the mean and standard deviations are chosen such that $\boldsymbol{\mu}_1(\mathbf{x}_0) = 0$ and $\sigma_1(\mathbf{x}_0) = 1$. We further require $\boldsymbol{\mu}_0(\mathbf{x}_0) = \mathbf{x}_0$ and $\sigma_0(\mathbf{x}_0) = \sigma_{\text{min}}$ where σ_{min} is small enough to not distort the data much. The corresponding vector field can then be written as

$$u_t(\boldsymbol{\psi}_t(\mathbf{x})|\mathbf{x}_0) = \frac{d\boldsymbol{\psi}_t(\mathbf{x})}{dt} \tag{4.72}$$

which yields to the CFM loss for Gaussian trajectories:

$$L_{\text{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\mathbf{x}_0\sim p_{\text{data}}(\mathbf{x}_0),\mathbf{x}_1\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\left\|\mathbf{v}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\mathbf{x}_1), t) - \frac{d\boldsymbol{\psi}_t(\mathbf{x}_1)}{dt}\right\|^2\right]. \tag{4.73}$$

Note that here $\boldsymbol{\psi}_t$ is conditioned on the data sample $\mathbf{x}_0$. The time dependent mean and standard deviation can be chosen to scale linearly in time, i.e. $\boldsymbol{\mu}_t(\mathbf{x}) = t\mathbf{x}_0$ and $\sigma_t(\mathbf{x}) = 1 - (1 - \sigma_{\text{min}})t$. Inserting these choices into the flow Equation 4.71 to substitute the second term in Equation 4.73 finally yields the optimal transport CFM loss that can be used to effectively train $\mathbf{v}_{\boldsymbol{\theta}}$:

$$L_{\text{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1}\left[\|\mathbf{v}_{\boldsymbol{\theta},t}(\boldsymbol{\psi}_t(\mathbf{x}_1)) - (\mathbf{x}_0 - (1 - \sigma_{\text{min}})\mathbf{x}_1)\|^2\right] \tag{4.74}$$

where $\mathbf{x}_0$ are samples from the data distribution and $\mathbf{x}_1$ are samples from a standard normal distribution.

Since the mean and standard deviation interpolate linearly between the data and Gaussian space, the conditional flow $\boldsymbol{\psi}_t$ is the *optimal transport displacement map* between the two Gaussians. Data points that are transformed under such a map move along a straight line trajectory at constant speed. This leads to particular stable and fast training of the CNF model compared to the above discussed diffusion models where the noise scheduler leads to curved trajectories (as can be seen in Figure 4.8). Empirically, the straight paths also allow for faster sampling at the same fidelity or higher fidelity at the same sampling speed [212, 224].

Once trained, we can generate new samples with various black-box ODE solvers, where the simplest fixed-step solver is the Euler solver given by Equation 4.38. We employ flow matching for training the EPiC-FM model for jet generation in Chapter 6.

4.5 Consistency Models

Consistency Models [219] are the newest type of generative models discussed in this chapter. They are based on the idea of learning a consistent mapping of a probability path between data space and a simple distribution such that the model can map from any point of this trajectory to the data space. An important property of the model is *self-consistency*, i.e. any point on the trajectory is mapped to the same initial point in data space. This allows for

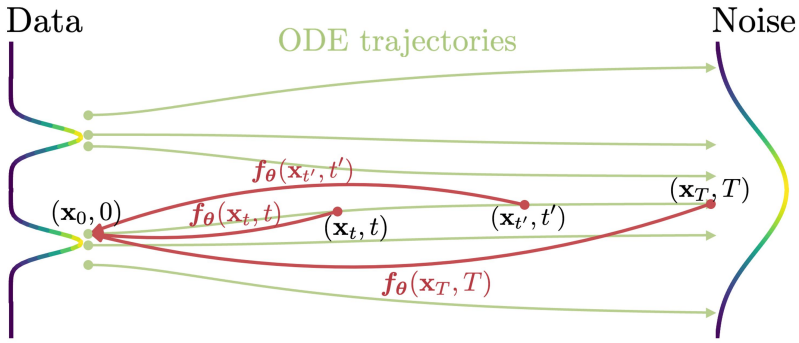


Figure 4.9: The probability flow ODE trajectories between data space and a simple noise distribution which are modeled by a consistency model $\mathbf{f}_{\boldsymbol{\theta}}$. The consistency model maps any point on the trajectory to the data space. Figure taken from Reference [219].

both single step generation, i.e. by sampling from the simple distribution and evaluating the consistency model once to generate a new data sample, but it allows also for multi-step generation since iteratively samples can be denoised, "renoised", and denoised again. Hence, with consistency models one can trade between fast single-step generation and potentially more accurate multi-step generation. A consistency model can either be trained directly from data, or it can be distilled from an already trained diffusion model. By distilling a diffusion model into a consistency model one can achieve single-step generation with similar generative fidelity as the diffusion model achieves with many model evaluations. In the following the training and distillation process are introduced in accordance with Reference [219].

The theory behind consistency models is based on the probability flow ODE Equation 4.66. Considering a unique solution trajectory $\{\mathbf{x}_t\}_{t\in[\epsilon,T]}$ of the ODE, the *consistency function* is defined as $\mathbf{f} : (\mathbf{x}_t, t) \rightarrow \mathbf{x}_\epsilon$ where time step $t = \epsilon$ denotes the minimum data perturbation such that $\mathbf{x}_\epsilon \approx \mathbf{x}_0$. The consistency function is self-consistent such that evaluating it at any point yields the same output, i.e. $\mathbf{f}(\mathbf{x}_t, t) = \mathbf{f}(\mathbf{x}_{t'}, t')$. A consistency model $\mathbf{f}_{\boldsymbol{\theta}}$ with trainable parameters $\boldsymbol{\theta}$ is trained to estimate the consistency function by utilizing the self-consistency property. This idea is visualized in Figure 4.9.

Once trained, we can sample from the simple distribution, i.e. $\hat{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, T^2\mathbf{I})$ where T is the maximum time step like in diffusion models, and generate new data samples $\hat{\mathbf{x}}_\epsilon$ by evaluating the consistency model:

$$\hat{\mathbf{x}}_\epsilon = \mathbf{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_T, T). \tag{4.75}$$

Hence, generation is performed in a single model evaluation step just like in a GAN. Additionally, multi-step generation is possible by first evaluating $\hat{\mathbf{x}}_\epsilon = \mathbf{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_T, T)$, then sampling another noise vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, using that noise to produce a "renoised" sample for time step t' along the ODE trajectory as $\hat{\mathbf{x}}'_t = \hat{\mathbf{x}}_\epsilon + \sqrt{t'^2 - \epsilon^2}\mathbf{z}$ and finally evaluating the consistency model again to generate an (ideally) higher fidelity data sample $\hat{\mathbf{x}}'_\epsilon = \mathbf{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t'}, t')$. This would be a two-step generation process, but it can be expanded to arbitrary many steps with $\epsilon < t'_1 < t'_2 < \cdots < T$ where time steps t'_i are found by a greedy algorithm that maximizes the final generative fidelity. The overall single- and multi-step generation procedure is illustrated in Figure 4.10.

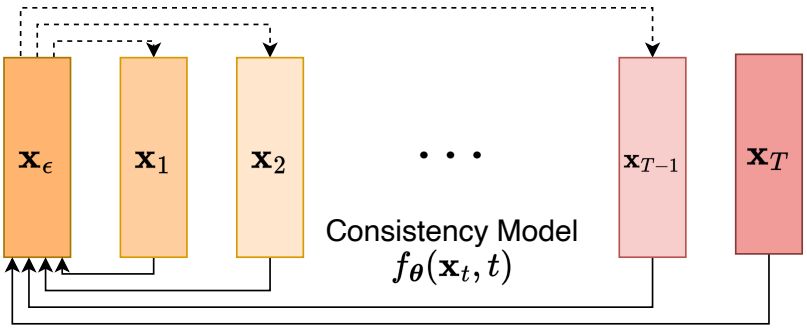


Figure 4.10: Illustration of the single- and multi-step generation process with a consistency model.

A consistency model can be either from a trained diffusion model or directly from data. Here first is the distillation approach discussed, known as *consistency distillation* (CD). As ingredients for the distillation we consider the score model $\mathbf{s}_\phi$ plugged into the probability flow ODE Equation 4.66 and a numerical ODE solver to evaluate the ODE trajectory. We discretize the time steps in the interval $[\epsilon, T]$ into $N-1$ steps as $t_1 = \epsilon < t_2 < \cdots < t_N = T$. With sufficiently large N, we assume $\mathbf{x}_{t_n} \approx \mathbf{x}_{t_n+1}$ when running the numerical ODE solver one step from t_{n+1} to t_n. Specifically, the sample $\hat{\mathbf{x}}^\phi_{t_n}$ produced by the update function $\boldsymbol{\Phi}(\cdots;\boldsymbol{\phi})$ of a single-step ODE solver applied to the ODE is given by

$$\hat{\mathbf{x}}^\phi_{t_n} = \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\boldsymbol{\Phi}(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi}). \tag{4.76}$$

When using the Euler solver given by Equation 4.38, i.e. $\boldsymbol{\Phi}(\mathbf{x}, t; \boldsymbol{\phi}) = -t\mathbf{s}_\phi(\mathbf{x}, t)$, the update function simplifies to

$$\hat{\mathbf{x}}^\phi_{t_n} = \mathbf{x}_{t_{n+1}} - (t_n - t_{n+1})t_{n+1}\mathbf{s}_\phi(\mathbf{x}_{t_{n+1}}, t_{n+1}). \tag{4.77}$$

A sample $\mathbf{x}_{t_{n+1}}$ along the ODE trajectory can be generated by first sampling $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ and then corrupting it with Gaussian noise, i.e. $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}, t^2_{n+1}\mathbf{I})$.

This way two adjacent points $(\hat{\mathbf{x}}^\phi_{t_n}, \mathbf{x}_{t_{n+1}})$ on the ODE trajectory can be created — one from data and the other from one step of the ODE solver. Due to the self-consistency requirement, the loss function of the consistency model is given by a distance such as MSE between the two points:

$$L_{\text{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{x}, n, \mathbf{x}_{\mathbf{t_{n+1}}}}\left[\lambda_{t_n} \|\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}) - \mathbf{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^\phi_{t_n}, t_n)\|^2\right] \tag{4.78}$$

where $n \sim U(1, N-1)$, λ_t is a time-dependent weighting function usually set $\lambda_t = 1$, and $\boldsymbol{\theta}^-$ is a running average of the previous values of $\boldsymbol{\theta}$. For the optimization of $f_{\boldsymbol{\theta}}$ the CD loss is minimized with respect to $\boldsymbol{\theta}$ while $\boldsymbol{\theta}^-$ is updated with an exponential moving average (EMA). Using common distillation terminology, we can call $\mathbf{f}_{\boldsymbol{\theta}^-}$ the "target model", $\mathbf{f}_{\boldsymbol{\theta}}$ the "student model", and $\mathbf{s}_\phi$ the "teacher model". Ideally, the loss converges and $\boldsymbol{\theta} = \boldsymbol{\theta}^-$ is achieved, i.e. after sufficiently long training the student model is equivalent to the target model. A flow chart of the overall CD procedure is shown in Figure 4.11.

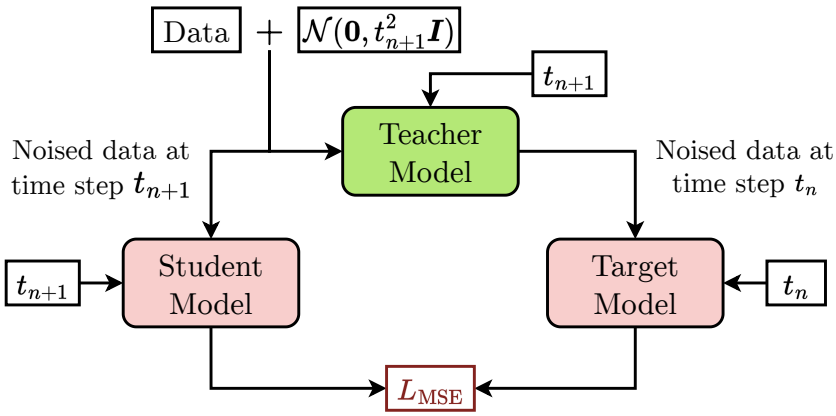


Figure 4.11: Overview of the consistency distillation performed to distill a diffusion model (teacher) into a consistency model (student and target model). The student model weights are updated via gradient descent, while the target model weights are an exponential moving average of the student model weights. Figure originally published in Reference [5].

The alternative method of training a consistency model directly is called *consistency training* (CT). Instead of approximating the score with $\mathbf{s}_\phi$ one can make use of the following unbiased estimator of the score [219]:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}, \mathbf{x}_t \sim \mathcal{N}(\mathbf{x}; t^2\mathbf{I})} \left[\frac{\mathbf{x}_t - \mathbf{x}}{t^2} \right]. \tag{4.79}$$

In CT, this estimator replaces the score model that is used for CD. Assuming the Euler method for the ODE solver, the CT loss function is given by

$$L_{\text{CT}}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = \mathbb{E}_{\mathbf{x},n,\mathbf{z}} \left[\lambda_{t_n} \| \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}) - \mathbf{f}_{\boldsymbol{\theta}^-}(\mathbf{x} - t_n\mathbf{z}, t_n) \|^2 \right] \tag{4.80}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $N \to \infty$. In practice, the CT loss is implemented with a progressively increasing number of steps N to approximate the continuous limit of $\Delta t \to 0$. Like in CD, the CT loss is minimized with respect to $\boldsymbol{\theta}$ while $\boldsymbol{\theta}^-$ is updated with an EMA.

Empirically, Reference [219] shows that using CD leads to more accurate consistency models than CT. In subsequent work [225], suggestions for an improved parameterization of CT are made that include alternative distance measures other than MSE in the CT loss, an alternate weighting function, the removal of the EMA update of the target model, and a changed noise and time step scheduler. We employ consistency distillation in Chapter 6 to distill the diffusion model CaloClouds II into a consistency model dubbed CaloClouds II (CM) for calorimeter shower generation. This greatly speeds up the generation process while maintaining high generative fidelity.

Part II

Developed Models and Their Evaluation

Chapter 5

Voxelized Modeling of Calorimeter Showers

The results presented in this chapter have been previously published in Reference [1] in collaboration with Sascha Diefenbacher, Engin Eren, Frank Gaede, Gregor Kasieczka, Anatolii Korol, and Katja Krüger. It is a more detailed analysis of the bounded information bottleneck autoencoder (BIB-AE) introduced in Reference [6] by the same authors. My contributions to this research were hyperparameter studies of the BIB-AE model, analyzing the physics encoded in the latent space of the BIB-AE, improving the BIB-AE with a kernel density estimation of the latent space, writing the majority of Reference [1], and presenting the results at the *25th International Conference on Computing in High-Energy and Nuclear Physics* (vCHEP 2021). Additional results discussed in this chapter on the evaluation of generative models were obtained under my supervision as part of the bachelor theses of Jan Schreiber [17] and Nana Marie Werther [16]. The figures and tables in this chapter are similar or identical to the ones published in References [1, 6, 16, 17].

To compare theory predictions with measurements at collider experiments a lot of simulated events are necessary. With increasing luminosity and data collection capabilities of detector systems at future colliders, the need for larger simulated datasets increases as well. To keep the computational cost sustainable, novel fast simulation techniques are being developed. In particular, high-granularity calorimeter systems require high-fidelity fast simulation methods. A very promising method for such fast simulations is using generative deep learning models. This was pioneered with the *CaloGAN* in References [39–41] for calorimeters with a regular granularity similar to the one found in the current CMS and ATLAS detectors. The work presented here builds upon the introduction of a high fidelity generative model for the fast simulation of high-granularity calorimeter showers.

To study the performance of the generative model, we use it to simulate electromagnetic showers in the electromagnetic calorimeter (ECAL) of the envisioned International Large Detector (ILD). Since the cells in the ILD calorimeter system are rectangular, the showers can be conveniently represented as a fixed-grid 3-dimensional image with each voxel representing a sensor cell of the calorimeter. This way we can use a convolutional neural network (CNN) architecture and common machine learning frameworks to create high-fidelity models for fast

shower simulation. A downside of this voxelized representation of high-granularity showers is that the images are rather sparse, leading to many empty voxels on the cost of computation efficiency. This limitation is overcome in subsequent work discussed in Chapter 6.

Based on the information bottleneck (IB) principle [226] and as a unification of autoencoder and generative adversarial network variants, the bounded information bottleneck autoencoder (BIB-AE) was introduced in Reference [199]. A theoretical introduction to the BIB-AE is given in Section 4.2.3. In Reference [6], we introduce a BIB-AE implementation that can produce high-granularity calorimeter showers with high fidelity at a high speed.

In this chapter, the latent space of a BIB-AE — trained to generate high-granularity photon calorimeter showers — is explored to interpret what physical information is encoded in its latent space. This understanding helps in optimizing the hyperparameters of the model, allows for improved latent space sampling, offers the possibility of targeted sampling, and increases the trust in the model's capability. We further explore evaluation metrics to better benchmark the performance of calorimeter shower generative models.

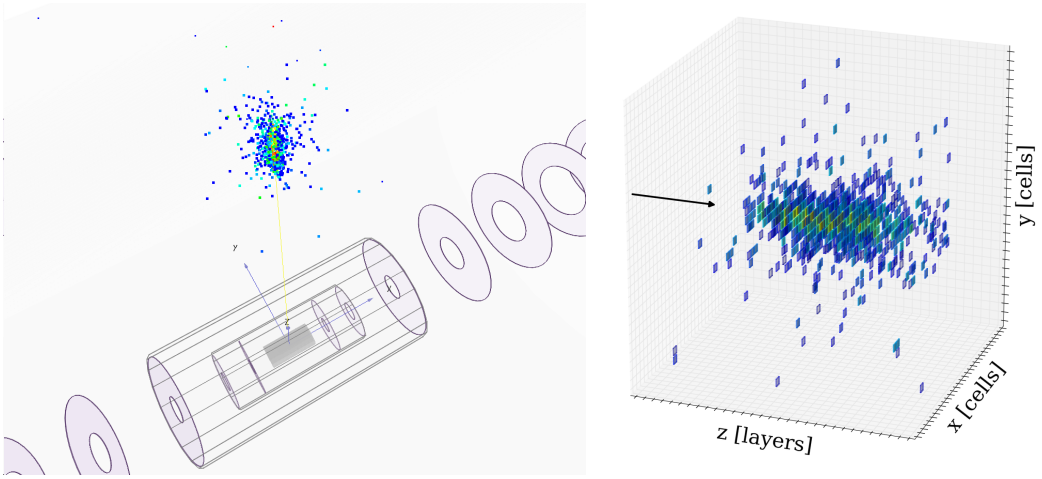
In a follow-up publication, we applied the same BIB-AE model — with slight modifications — to hadronic showers of charged pions in the ILD HCAL. We refer to Reference [9] for details, since in this thesis the focus lies on the generative modeling of electromagnetic showers. The BIB-AE was further applied to photon showers with a non-perpendicular incident angle in Reference [51].

In Section 5.1 we introduce the dataset of simulated photon showers and their voxelization into 3D images. The BIB-AE model is introduced in Section 5.2 and its training and sampling methods are discussed. In Section 5.3 we analyze the latent space of the BIB-AE model and in Section 5.4 we apply the understanding of the latent space to improve the generative fidelity of the BIB-AE. We conclude with studies of several evaluation metrics for generative models in Section 5.5 and the introduction of a novel evaluation measure, the Fréchet regression distance (FRD), in Section 5.6. A summary is provided in Section 5.7.

5.1 Calorimeter Shower Images

For this study, a large number of photon showers in the Si-W ECAL of the proposed ILD were simulated[1]. The ECAL is a sampling calorimeter with 30 active silicon layers and 30 passive tungsten layers — with the first 20 having a thickness of 2.1 mm and the following 10 a thickness of 4.2 mm. The silicon sensors have a size of 5×5 mm^2 and a thickness of 0.525 mm. In the lateral $x - y$ plane, we consider a 30×30 cells section of the ECAL (for each of the 30 layers). Together with the z-axis in the longitudinal direction (the incident shower direction and direction of shower development), this allows us to represent the photon showers in a $30 \times 30 \times 30$ regular grid. We will be referring to this data tensor $x \in \mathbb{R}^{30\times30\times30}$ as a 3-dimensional *calorimeter image*, where each voxel (a 3D pixel) corresponds to the energy deposited in a specific cell sensor. Depending on the specific layer, the cells are slightly staggered in the x-direction (but uniform in the y-direction), hence slight irregularities are introduced by moving the pixel positions to a cubic 3D image.

[1]The simulated data was created by Engin Eren.



(a) ILD ECAL geometry **(b)** 50 GeV GEANT4 shower

Figure 5.1: (a) Visualization of a GEANT4 simulated photon shower in the ILD ECAL with an incident energy of 60 GeV. For reference, the cylindrical structure of the tracker is shown. Figure taken from Reference [6]. **(b)** 3D image of a 50 GeV GEANT4 shower. The arrow shows the direction of the incident photon.

The ILD simulation runs in the ILCSOFT [227] framework. For the full simulation of photon showers in the ECAL, GEANT4 [118] Version 10.4 with the QGSP-BERT physics list is used. The realistic detector is implemented in DD4HEP [117] Version 1.11.

All incident photons are aimed at the center of the $x-y$ lateral plane of the barrel ECAL section. A visualization of a photon shower in the ILD ECAL is shown in Figure 5.1. A total of 950,000 photon showers with uniformly distributed incident energy between 10 and 100 GeV are simulated for the training set. Multiple datasets are simulated for evaluation of the generative models: 40,000 uniformly distributed showers for the evaluation of the full (10-100 GeV) energy spectrum as well as 4,000 single energy showers for energies between 20 and 90 GeV in discrete 10 GeV steps. Further details on the dataset can be found in Reference [6].

5.2 Bounded Information Bottleneck Autoencoder (BIB-AE)

In Reference [6] we introduce the bounded information bottleneck autoencoder (BIB-AE) as a generative model for the generation of photon calorimeter showers. We find an increased generative fidelity with the BIB-AE model when comparing it to a GAN and a Wasserstein-GAN (WGAN). The BIB-AE implementation[2] is inspired by Reference [199] and is a unification of multiple (variational) autoencoder and GAN variations such as VAE [193], β-VAE [195], adversarial autoencoder (AAE) [196], GAN [228], WGAN [190], WGAN-GP [191] and VAE-GAN [202]. For a theoretical introduction of the BIB-AE, see Section 4.2.3.

[2]The BIB-AE was implemented by Sascha Diefenbacher. A detailed explanation of the BIB-AE components can be found in his PhD thesis [153].

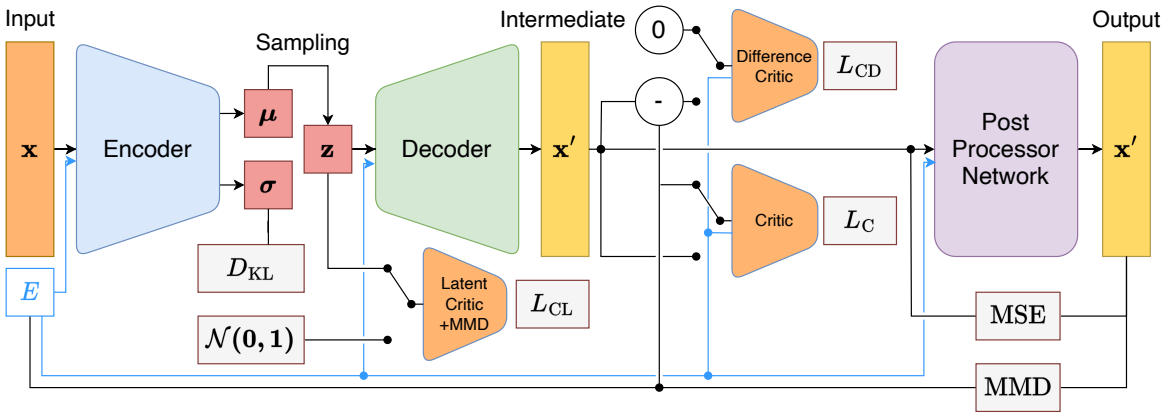


Figure 5.2: Schematic of the BIB-AE model including the Post Processor network. The model consists of multiple sub-models: An encoder, a decoder, two reconstruction critics, a latent space critic, and a Post Processor Network. The BIB-AE is conditioned on the energy of the incident particle E (blue lines). Figure adapted from Reference [6].

5.2.1 BIB-AE Model

An overview of the complete BIB-AE model framework is shown in Figure 5.2. The blue line indicates, that all sub-models are conditioned on particle incident energy E_{inc}. At the core of the model is a VAE, with the *Encoder* and *Decoder* model components. The Encoder encodes a 3D calorimeter image $\boldsymbol{x}$ in the latent space parameterized by the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ vectors. The Encoder network is implemented as a CNN using 3D convolutional layers intertwined with layer normalization [229] and a fully connected network stem. LeakyReLU [230] is used for all activation functions in the BIB-AE. Using the reparameterization trick (see Section 4.2.1) a latent space vector $\boldsymbol{z}$ is sampled. To regularize the latent space towards a normal Gaussian, the Kullback-Leibler divergence (KLD) is used as a loss function (scaled with a hyperparameter with respect to the reconstruction loss terms, just like a β-VAE [195]). For an additional latent space regularization, we employ a WGAN-like *Latent Critic* (like in an AAE [196]) and a maximum mean discrepancy (MMD) [200] loss. The latent critic is implemented as a simple fully connected network.

During training, the Decoder is used to reconstruct the encoded calorimeter images based on the latent vector $\boldsymbol{z}$. It is implemented as a reverse CNN and largely inverts the network architecture of the Encoder. The model stem is a fully connected network, followed by 3D transpose convolutional layers intertwined with layer normalization used for further up-sampling, and additional regular 3D convolutional layers with layer normalization. The output of the Decoder is a full 3D image in the same data format as the original training data. Once trained the Decoder is used to generate novel calorimeter showers from a randomly sampled $\boldsymbol{z}$.

Instead of an MSE reconstruction loss, we employ two Wasserstein critic networks to optimize the shower generation by the encoder-decoder. Just like in a WGAN, the *Critic* learns the difference between GEANT4 and generated/reconstructed showers. As it is a distributional loss, i.e. it minimizes the difference between the distribution of (a batch of)

GEANT4 showers vs. the distribution of (a batch of) reconstructed showers, we need an additional critic that directly compares individual showers, namely the encoded shower with the decoded/reconstructed shower. For this purpose, we employ a *Difference Critic* which is used to minimize the (shower-wise) distance between an input and a reconstructed shower. Through adversarial training and the right balancing of loss terms, the Encoder and Decoder improve as the Critic and Difference Critic improve simultaneously. The critics are trained using gradient penalty [191] and weight updates are alternated between iterations with either critic training or encoder-decoder training. In practice, both critics are implemented in a single neural network with a single critic output (the learned Wasserstein distance). The Critic part of the network is implemented as a CNN with 3D convolutional layers intertwined with layer normalization and a fully connected network stem. The Difference Critic part is a fully connected network. The FCN stems of both critics are combined to allow for a single output.

The final model part is the *Post Processor Network* which refines the reconstructed/generated shower. It consists of a pixel-wise neural network (3D convolutional layers with kernel size = (1,1,1) with layer normalization [229]) effectively able to slightly adjust each pixel. This allows the model to i.e. achieve a very good representation of the cell energy spectrum including the precise modeling of the MIP (minimum ionizing particle) peak. The model is trained using both MSE and MMD loss functions. The MSE loss is applied pixel-wise, while the MMD loss is used to compare the distribution of pixel values. Since computing the MMD for up to 27,000 pixels is prohibitively expensive, we implemented a *Sorted-Kernel-MMD*. For this MMD variant, we first sort all pixel values for both generated and GEANT4 showers and then apply the MMD on subsets of the 100 pixels using 25 moving comparison windows. This comparison includes an adequate number of total hits, as anticipated above the half-MIP energy threshold.

The whole BIB-AE model has about 71M weights, of which the Decoder (the actual generator) takes up about 35M weights — largely attributed to the use of fully connected layers in the stem of the Decoder before applying 3D convolutions. The BIB-AE model was implemented using PYTORCH [173].[3]

5.2.2 BIB-AE Training

To train the overall BIB-AE model, we update the Encoder and Decoder networks, while keeping the weights of the critic networks fixed, and vice-versa. We define the Encoder $\mathbf{e}$, the Decoder $\mathbf{d}$, the Critic $\mathbf{c}$, the Latent Critic $\mathbf{c}_{\mathrm{L}}$, the Difference Critic $\mathbf{c}_{\mathrm{D}}$, and the Post Processor Network $\mathbf{n}_{\mathrm{PP}}$. Overall the loss during training of the BIB-AE model (without the Post Processor Network) is a weighted summation of the reconstruction losses (Critic and Difference Critic) and the latent space regularization losses (reverse D_{KL}, MMD, and Latent

[3]The BIB-AE model is available at: `https://github.com/FLC-QU-hep/getting_high`

Critic):

$$\begin{aligned} L_{\text{BIB-AE}} = &- \beta_{\mathbf{c}} \cdot \mathbb{E}_{\boldsymbol{x}}[\mathbf{c}(\mathbf{d}(\mathbf{e}(\boldsymbol{x})))] \\ &- \beta_{\mathbf{c}_\text{D}} \cdot \mathbb{E}_{\boldsymbol{x}}[\mathbf{c}_\text{D}(\mathbf{d}(\mathbf{e}(\boldsymbol{x})) - \boldsymbol{x})] \\ &- \beta_{\mathbf{c}_\text{L}} \cdot \mathbb{E}_{\boldsymbol{x}}[\mathbf{c}_\text{L}(\mathbf{e}(\boldsymbol{x}))] \\ &+ \beta_{\text{KLD}} \cdot D_{\text{KL}}(\boldsymbol{z}||\mathbf{e}(\boldsymbol{x})) \\ &+ \beta_{\text{MMD}} \cdot \text{MMD}(\mathbf{e}(\boldsymbol{x}), \boldsymbol{z}), \end{aligned} \tag{5.1}$$

with hyperparameters β for weighting each loss contribution, $\boldsymbol{x}$ sampled from the data distribution, and $\boldsymbol{z}$ sampled from normal distributed noise $\mathcal{N}(\mathbf{0}, \mathbf{I})$. For simplicity, the explicit conditioning of all sub-models on the incident energy E_inc is omitted.

While training the critic networks, the Encoder and Decoder weights remain fixed. Following the definition of the WGAN loss with gradient penalty [191], the loss L_C for critic c is given by:

$$L_\text{C} = \mathbb{E}_{\boldsymbol{x}}[\mathbf{c}(\mathbf{d}(\mathbf{e}(\boldsymbol{x})))] - \mathbb{E}_{\boldsymbol{x}}[\mathbf{c}(\boldsymbol{x})] + \lambda\, \mathbb{E}_{\hat{\boldsymbol{x}}}[(\parallel \nabla_{\hat{\boldsymbol{x}}} \mathbf{c}(\hat{\boldsymbol{x}}) \parallel_2 -1)^2], \tag{5.2}$$

with $\hat{\boldsymbol{x}}$ being an equal mixture of samples drawn from the data distribution and reconstructed data $\mathbf{d}(\mathbf{e}(\boldsymbol{x}))$, and λ as a weighting hyperparameter for the gradient penalty term (typically set to 10). The loss for the Difference Critic $\mathbf{c}_\text{D}$ is similarly given by:

$$\begin{aligned} L_\text{CD} =& \mathbb{E}_{\boldsymbol{x}}[\mathbf{c}_\text{D}(\mathbf{d}(\mathbf{e}(\boldsymbol{x})) - \boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}}[\mathbf{c}_\text{D}(\boldsymbol{x} - \boldsymbol{x} = \mathbf{0})] \\ &+ \lambda\, \mathbb{E}_{\hat{\boldsymbol{x}}}[(\parallel \nabla_{\hat{\boldsymbol{x}}} \mathbf{c}_\text{D}(\hat{\boldsymbol{x}}) \parallel_2 -1)^2]. \end{aligned} \tag{5.3}$$

Finally, the Latent Critic loss L_CL is given by:

$$L_\text{CL} = \mathbb{E}_{\boldsymbol{x}}[\mathbf{c}_\text{L}(\mathbf{e}(\boldsymbol{x}))] - \mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,1)}[\mathbf{c}_\text{L}(\boldsymbol{z})] + \lambda\, \mathbb{E}_{\hat{\boldsymbol{x}}_{\boldsymbol{z}}}[(\parallel \nabla_{\hat{\boldsymbol{z}}_{\boldsymbol{x}}} \mathbf{c}_\text{L}(\hat{\boldsymbol{z}}_{\boldsymbol{x}}) \parallel_2 -1)^2], \tag{5.4}$$

with $\hat{\boldsymbol{z}}_{\boldsymbol{x}}$ being a equal mixture of the encoded data $\mathbf{e}(\boldsymbol{x})$ (including reparameterization) and normal distributed noise $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.

After training the BIB-AE, the Post Processor Network $\mathbf{n}_\text{PP}$ is trained using the Sorted-Kernel-MMD between the input and output calorimeter shower images and the MSE between the reconstructed and output images. While the MMD helps to align the hit energy spectrum directly with the original data distribution, the MSE guarantees that the modifications by the Post Processor Network are minimal. The loss function is implemented as:

$$L_\text{PP} = \text{MMD}(\mathbf{n}_\text{PP}[\mathbf{d}(\mathbf{e}(\boldsymbol{x}))], \boldsymbol{x}) + \beta_\text{MSE} \cdot \text{MSE}(\mathbf{n}_\text{PP}[\mathbf{d}(\mathbf{e}(\boldsymbol{x}))], \mathbf{d}(\mathbf{e}(\boldsymbol{x}))). \tag{5.5}$$

5.2.3 BIB-AE Sampling

During shower generation, only two model parts are used: The Decoder and the Post Processor Network. While during training data showers are reconstructed, during sampling new (unseen) showers are generated. For generating showers, a conditional incident energy is set and a random latent vector $\boldsymbol{z}$ is sampled from a normal Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$. $\boldsymbol{z}$ is decoded by the Decoder and the generated shower is refined by the Post Processor Network. A detailed comparison of the BIB-AE fidelity in comparison to a GAN and a WGAN is given

in Reference [6]. Generating a shower with GEANT4 on a single CPU core takes (on average over the whole 10-100 GeV range) 4082 ± 170 ms, while the sampling with the BIB-AE takes 418.04 ± 0.20 ms on the same hardware (with batch size 100), resulting in a $10\times$ speed-up. Using a NVIDIA® V100 (32 GB) GPU, the speed-up is even greater with 1.42 ± 0.01 average sampling time of a single shower (with batch size 100), leading to a speed-up of $2874\times$. Note that the speed-up using CPUs in comparison to GEANT4 is of more practical importance than the GPU timing since GPUs are much more expensive and not as widely available. Additionally, GEANT4 is currently only available for CPU hardware.

An advancement over the baseline BIB-AE introduced in Reference [6] constitutes the introduction of the *latent kernel density estimation* (KDE) in Reference [1]. This multi-dimensional KDE estimates the distribution of the encoded (resampled) $\boldsymbol{z}$ latent space of the whole training dataset. During sampling, instead of $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ we now sample $\boldsymbol{z}$ from the KDE. This way correlations encoded in the latent space are replicated by the KDE sampling and a higher fidelity modeling is achieved. This simple idea to extend a VAE was previously used in publications such as Reference [27], where it is introduced as *density information buffer* for event generation. In subsequent work, such as References [50, 51], the KDE is replaced with a normalizing flow to allow for conditional latent space sampling. The latent space and the KDE are further discussed in the following Section 5.3.

More details on the BIB-AE model including the exact architectures and hyperparameters can be found in References [1, 6, 153]. With small changes, the BIB-AE was also used in subsequent work [9, 51].

5.3 Latent Space Decoding

In any autoencoder, some information about the original data is encoded into the latent space and then used for the reconstruction. After benchmarking the high generative fidelity and the substantial speed-up of generating showers with the BIB-AE compared to GEANT4 in Reference [6], we are exploring the latent space of the BIB-AE to understand if and what physics information is encoded (learned) by the model. This understanding of the latent space can be used either for targeted sampling — of photon showers with specific physical properties — or to improve the overall fidelity when sampling from the BIB-AE, i.e. by optimizing hyperparameters or by using the above-mentioned KDE sampling. A connection of this latent space study can be drawn to similar studies performed with autoencoders trained on photos. Here, researchers are able to interpolate between specific latent variables to change specific aspects of generated photos, i.e. the rotation of objects or faces [231].

5.3.1 Latent Space Size

As a baseline, the BIB-AE model in Reference [6] is implemented with $n \equiv \dim(\boldsymbol{z}) = 24$ trainable latent variables $\mu_{i\in[1,n]}$ and $\sigma_{i\in[1,n]}$. Additionally, $m = 488$ (not trainable) latent space variables are sampled from a normal Gaussian $\mathcal{N}(0, 1)$. Hence the total dimensionality of the latent vector $\boldsymbol{z}_{\text{total}}$ is $\dim(\boldsymbol{z}_{\text{total}}) = n + m$. To understand the optimal size of the trainable latent space $\boldsymbol{z}$, we vary n while keeping $\dim(\boldsymbol{z}_{\text{total}}) = 512$ constant (for the calculation of the

BIB-AE loss in Equation 5.1 only the n trainable latent variables are considered). The size of the latent space should be explored in relation to its information content, since even a low number of latent variables could contain more information than a large number.

We use the Kullback-Leibler divergence (KLD) to measure the difference between the distribution of the latent space $\boldsymbol{z}$ and a normal Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Note we refer here technically to the reverse Kullback-Leibler divergence, which is typically used in VAE trainings. The KLD of a single set of latent variables μ_i and σ_i compared to a normal Gaussian distribution is in practice given by [193]:

$$D_{\mathrm{KL},i} = -\frac{1}{2}\left(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2\right), \tag{5.6}$$

typically calculated with the natural logarithm. The total D_{KL} of the latent space (the same as used as part of the BIB-AE loss function) is the summation over the $D_{\mathrm{KL},i}$ of all latent variables i, $D_{\mathrm{KL}} = \sum_{i=1}^{n} D_{\mathrm{KL},i}$. With the KLD, we can quantify how much information is encoded in the latent space and each latent variable i individually [232,233]. Depending on the logarithm in Equation 5.6, one unit of information is typically measured in *nats* (natural units of information, based on the natural logarithm), or *bits* (binary units of information, using the base-2 logarithm). Latent variables sampled from a normal Gaussian $\mathcal{N}(\mu_i = 0, \sigma^2 = 1)$ contain no information (0 nats (bits)), while latent variable z_i sampled with $\mu_i \neq 0$ and/or $\sigma_i^2 \neq 1$ contains information $D_{\mathrm{KL},i} > 0$ nats.

By adjusting the hyperparameter β_{KLD} in Equation 5.1 as well as by changing n, the amount of information encoded in the latent space is steered. By reducing β_{KLD} with a fixed and large enough n, the information content is increased as the KLD term is lower weighted in the overall loss allowing for a higher KLD. For a fixed β_{KLD} and increasing n, the information content is increased as more variables are available to contain information. The information content however saturates around a certain n once a KLD value is reached that is bound by the β_{KLD} weight in comparison to the other loss terms.

We demonstrate this behavior of the latent space scaling by training multiple models with various $n \in [2, 512]$ and fixed $\beta_{\mathrm{KLD}} = 0.05$ (the default value in Reference [6]), Note that for $n = 24$ the training from Reference [6] was used. For comparison, we also trained one model with a larger $\beta_{\mathrm{KLD}} = 0.4$ and a fixed $n = 24$. In Figure 5.3 we show the performance of the resulting KLD in this study. In both figures, the individual latent variables per model are sorted with descending KLD (x-axis). In the left figure, we show the integrated KLD, i.e. the sum of all individual $D_{\mathrm{KL},i}$ up to and including latent variable i. We see, that with increasing n, the total KLD of the latent space of a model increases until it saturates above 60 latent variables and about 45 nats ($\approx$ 64 bits). Hence, at $\beta_{\mathrm{KLD}} = 0.05$, more than 60 trainable latent variables do not increase the information encoded in the latent space. If $n > 64$ variables are available, the remaining ones contain little to no information.

This can be observed in Figure 5.3 (right), where we show the KLD of the individual latent variables (sorted on the x-axis with decreasing KLD). For all models, a similar pattern emerges: two variables contain a much larger amount of information than the remaining ones, and at around latent variable 60, the KLD drops off to $D_{\mathrm{KL}} < 0.3$ nats. Note that for the model with $\beta_{\mathrm{KLD}} = 0.4$, only 7 latent variables contain information and the information saturates at $D_{\mathrm{KL}} \approx 5$ nats.

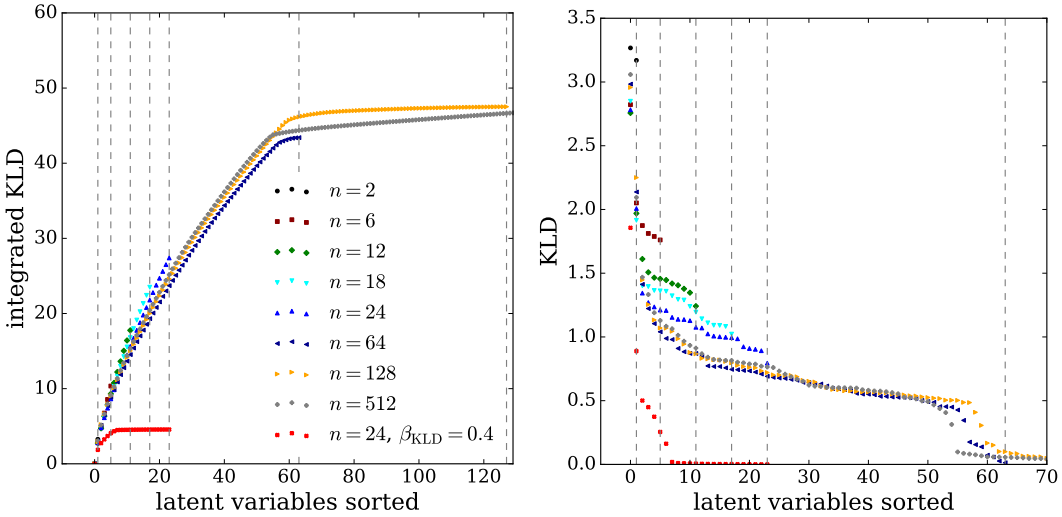


Figure 5.3: Left: Integrated Kullback-Leibler divergence (KLD) for latent variables sorted with decreasing KLD for BIB-AE models with various latent space sizes. **Right:** KLD of individual latent space variables sorted with decreasing KLD for BIB-AE models with various latent space sizes. The vertical dashed lines represent the latent space sizes investigated. All models are trained with a baseline weight $\beta_{\mathrm{KLD}} = 0.05$, except stated otherwise. Figures originally published in Reference [1].

Based on this assessment, we would naively expect that a model with about $n = 60$ yields optimal performance at $\beta_{\mathrm{KLD}} = 0.05$. To evaluate the models, we introduce the *fidelity score* S_{JSD} based on the Jensen-Shannon distance (JSD) [234] between GEANT4 and BIB-AE sampled showers in various 1D physical observables. The JSD is the square root of the Jensen-Shannon divergence D_{JS}, a symmetric version of the Kullback-Leibler divergence D_{KL}, given by:

$$D_{\mathrm{JS}}(P||Q) = \frac{1}{2}\left(D_{\mathrm{KL}}(P||M) + D_{\mathrm{KL}}(Q||M)\right) \tag{5.7}$$

where P and Q are the two distributions being compared and M is the mixture $M = \frac{1}{2}(P+Q)$.

As observables, we chose the ones closely investigated in Reference [6], namely the visible cell energy, the total visible shower energy for three individual incident energies, the number of cell hits (occupancy) for three individual incident energies, the center of gravity in z direction, the longitudinal energy profile, and the radial energy profile, i.e. the histograms presented in Figure 5.7. The total fidelity score S_{JSD} is calculated as a weighted summation of the S_{JSD} of each of the six distributions. Specifically, the weighting is done based on the JSD of each individual histogram compared to the JSDs of the other histograms (from the same model). The weighting is implemented as follows [1]:

1. Calculate the JSD of each histogram for each model and epoch that are compared in the overall score: $\mathrm{JSD}_{i,m,e}$ where $i \in [1, \ldots, 6]$ is the histogram index, m is the model index, and e is the epoch index.

2. The weighting factor for the JSD of each i histogram is given by the average $\overline{\mathrm{JSD}_{i,m,e}}$.

Table 5.1: Fidelity score S_{JSD} for the best epoch of multiple model configurations with various latent space sizes n. For $n = 24$ the best score out of multiple trainings is given. The mean score and standard deviation for those trainings is: $\overline{S}_{\text{JSD},24} = 1.02 \pm 0.12$. Only one training was performed for sizes $n \neq 24$. Table adapted from Reference [1].

latent size n	2	6	12	18	24	64	128	512
$S_{\textbf{JSD}}$	1.64	1.12	1.11	0.95	**0.83**	0.88	0.94	0.98

3. The final fidelity score S_{JSD} for each model m and epoch e is given by: $S_{\text{JSD},m,e} = \frac{1}{6} \sum_i \left[\text{JSD}_{i,m,e} \cdot (\overline{\text{JSD}_{i,m,e}})^{-1} \right]$.

The resulting fidelity scores for the best epochs of the different model configurations are presented in Table 5.1. We observe worse performance for very low n, the performance saturates for increasing n, and the best performance we observe for $n = 24$. At $n = 24$ we trained a total of seven models and show in the table the best-performing model with the lowest score. Over these seven training, we observe a mean and standard deviation of the fidelity score at $\overline{S}_{\text{JSD},24} = 1.02 \pm 0.12$. Due to computational limitation — the BIB-AE training takes up to 3 weeks on an NVIDIA® V100 GPU — the other models with $n \neq 24$ were only trained once. Yet, we observe that the original BIB-AE hyperparameter choice in Reference [6] is well motivated, a high generative fidelity can be reached with $n = 24$, and significantly larger latent space sizes do not offer a benefit. This further suggests, that the maximum amount of encoded information at 45 nats — encoded for example in the model with $n = 64$ — is not necessary for optimal generative fidelity.

5.3.2 Latent Space Analysis

Next, we investigate what kind of physics information is encoded in the latent space of the BIB-AE model. As we have seen, only a few latent variables contain most of the information. Therefore, we calculate the Pearson correlation coefficient r [235] between the five most information-rich latent variables $z_{i\in[0,4]}$ (the five highest KLD variables) as well as the conditioning E and various shower physics observables. As observables, we choose the 1st and 2nd moment in x-, y-, and z-direction $m_{i\in\{1,2\},j\in\{x,y,z\}}$ (the first moment is equivalent to the showers' center of gravity), the total visible energy E_{vis}, the particles incident energy E_{inc}, the number of hits n_{hits}, and the visible energy fraction in the first, second, and third third of the detector (in z-direction) $E_{i\in\{1,2,3\}}/E_{\text{vis}}$.

In Figure 5.4 we show these correlations for four model configurations: $n = 12$, 24, and 512, and for $n = 24$ either $\beta_{\text{KLD}} = 0.05$ (training from Reference [6]) or $\beta_{\text{KLD}} = 0.4$. Here, the encoded latent variables z_i are sampled from the encoding $\mathcal{N}(\mu_i, \sigma_i^2)$ ("reconstruction") and not from a normal Gaussian distribution $\mathcal{N}(0, 1)$ ("generation"). An expected perfect correlation of $r = 1.0$ is the correlation of the incident energy with itself, the visible energy, and the number of hits. An unexpected correlation is the very strong correlation of $r = 0.9$ between the most information-rich latent variable z_0 and the center of gravity in z-direction $m_{1,z}$. As the center of gravity in z is correlated to the shower start, it is also correlated to

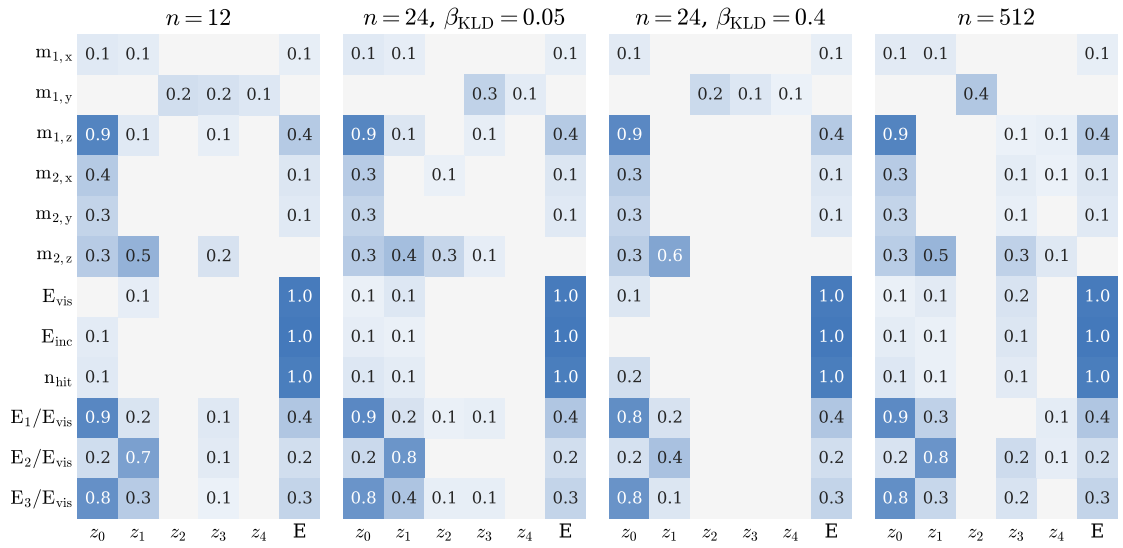


Figure 5.4: Pearson correlation coefficients between various physics observables and the latent space variables z_i of the five highest KLD latent variables (in descending order) as well as the incident particle energy E. Shown for BIB-AE models with latent size $n = 12$, 24, and 512. The baseline latent weight is $\beta_{KLD} = 0.05$ is used except for one training with $\beta_{KLD} = 0.4$ and $n = 24$. Zero Pearson correlation values are omitted. Figures originally published in Reference [1].

the total deposited energy in the first and third third of the detector, which shows a similarly strong correlation with z_0. The second most information-rich variable is correlated with $r \approx 0.5$ to the second moment in z $m_{2,z}$ and similarly correlated to the visible energy in the second third of the detector. Interestingly, these correlations are apparent in all four models, regardless of latent size n and KLD weight β_{KLD}. Hence, we can conclude that the BIB-AE always learns to encode the center of gravity in z-direction (and other associated observables such as the shower start) in the highest KLD variable. The correlation between z_0 and $m_{1,z}$ is also much stronger ($r = 0.9$) than between the conditioning energy E and $m_{1,z}$ ($r = 0.4$), which indicates that this is a very useful observable for the network to learn in addition to its conditioning.

Having identified this correlation, we can use the z_0 latent variable for targeted sampling of showers with a certain $m_{1,z}$. We demonstrate this in Figure 5.5, where we visualize five generated showers as a 3D image. These showers were sampled with the model from Reference [6] trained with $n = 24$ and $\beta_{\text{KLD}} = 0.05$ and all latent variables $z_{i\neq 0} = 0$ and values between -3 and 3 for z_0. We observe that with increasing z_0 values the shower starts later in the calorimeter effectively shifting the center of gravity. This is further visualized in Figure 5.6, where we show the energy profile of these five showers and observe how the layer with the highest energy shifts by changing the value of z_0.

Not only does this knowledge of the encoded latent information allow for targeted sampling of calorimeter showers with certain physical properties, it also highlights the fact that indeed important information is encoded in the latent space. This information may be lost when

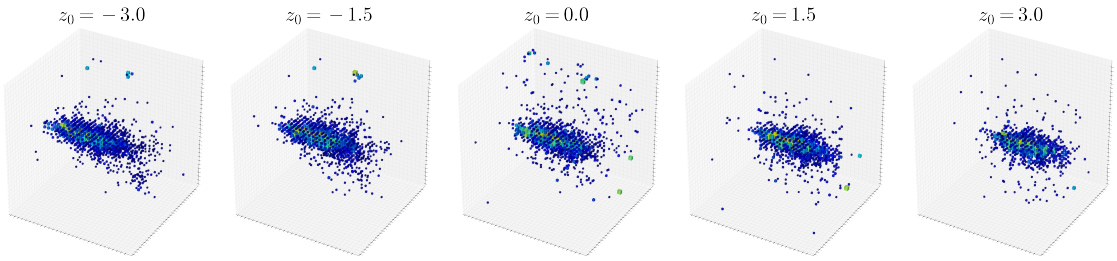


Figure 5.5: BIB-AE generated showers as 3D images. Generated from a latent space with all $z_{i>0} = 0$, except the highest KLD variable z_0 is varied between -3 and 3. The color coding corresponds to the cell energy. With increasing z_0 the shower start is moved to later in the calorimeter, since z_0 correlates with the center of gravity in z (shower incident axis). Figures originally published in Reference [1].

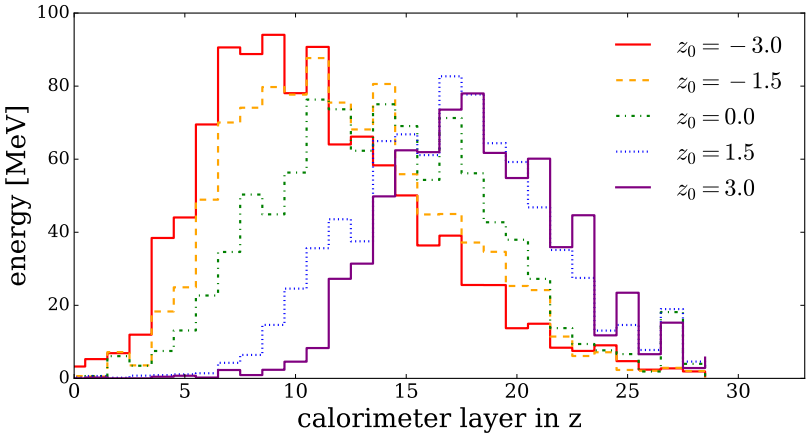


Figure 5.6: Longitudinal energy profile for multiple showers decoded from a latent space with all $z_{i>0} = 0$, except the highest KLD variable z_0 is varied between -3 and 3. Figure originally published in Reference [1].

generating new showers from a latent space simply sampled from $\mathcal{N}(0,1)$ — as is usually done in a VAE and our baseline BIB-AE setup — and might ultimately impact the achievable generative fidelity of the BIB-AE model.

5.4 Improved Generation with Latent Space Sampling

An alternative approach to sampling the latent space from a standard normal distribution $\mathcal{N}(0,1)$ is the sampling from a distribution that mirrors the encoded latent space during training and therefore can capture correlations between the latent variables that contain information. If each variable is sampled independently and identically distributed (i.i.d.) from $\mathcal{N}(0,1)$ these correlations are lost. Additionally, even independently the high KLD variables differ from normal distributed samples (otherwise they would contain $D_{\mathrm{KL},i} = 0$ nats of information) and are hence differently modeled during generation than during training by sampling them from $\mathcal{N}(0,1)$. Another sampling option would be to increase β_{KLD} weight thereby enforcing less information in the latent space. This way a sampling from $\mathcal{N}(0,1)$ is closer to the encoded latent space which could increase sampling fidelity. However, this requires a retraining of the BIB-AE model, while sampling from the encoded space can be performed with an already trained model.

5.4.1 Differential Distributions

We compare both GEANT4 and the baseline BIB-AE from Reference [6] with both options for improving the accuracy of the latent space sampling. For a model with a stronger regularized latent space, we retrain the BIB-AE with a higher $\beta_{\mathrm{KLD}} = 0.4$ than the baseline $\beta_{\mathrm{KLD}} = 0.05$. Additionally, we advance the BIB-AE model by sampling from a model of the encoded latent space, i.e. a kernel density estimate of the reparameterized $\boldsymbol{z}$ space. We implemented this by encoding 500,000 showers of the training set and applying a 25-dimensional KDE to model the latent space (25 dimensions, due to $n = 24$ and the energy conditioning). This way we can reuse the exact BIB-AE training done for Reference [6] (with $\beta_{\mathrm{KLD}} = 0.05$ and $n = 24$). Among other methods for sampling from the encoded space, this *KDE sampling* is also suggested in Reference [27] for the so-called *Buffer-VAE*.

In Figure 5.7 we show six differential distributions of shower observables: the visible cell energy (top left), the total visible energy for three fixed incident photon energies at 20, 50, and 80 GeV (top center), the number of hits for these three energies (top right), the center of gravity in z-direction (bottom left), the longitudinal energy profile (bottom center), and the radial energy profile (bottom right). All three models align well with GEANT4 in the cell energy distribution. Note that the edge around 0.2 MeV corresponds to the most probable energy deposition of a minimum ionizing particle (MIP) in a silicon sensor of the ILD Si-W ECAL at a perpendicular incident angle. This feature is particularly challenging to model and a good benchmark for a high-fidelity model such as the BIB-AE. The region below 0.1 MeV is greyed out as for all other shower observables a low energy cut is applied at this value since below about half a MIP the visible energy cannot be distinguished from the electronic noise of the silicon sensor.

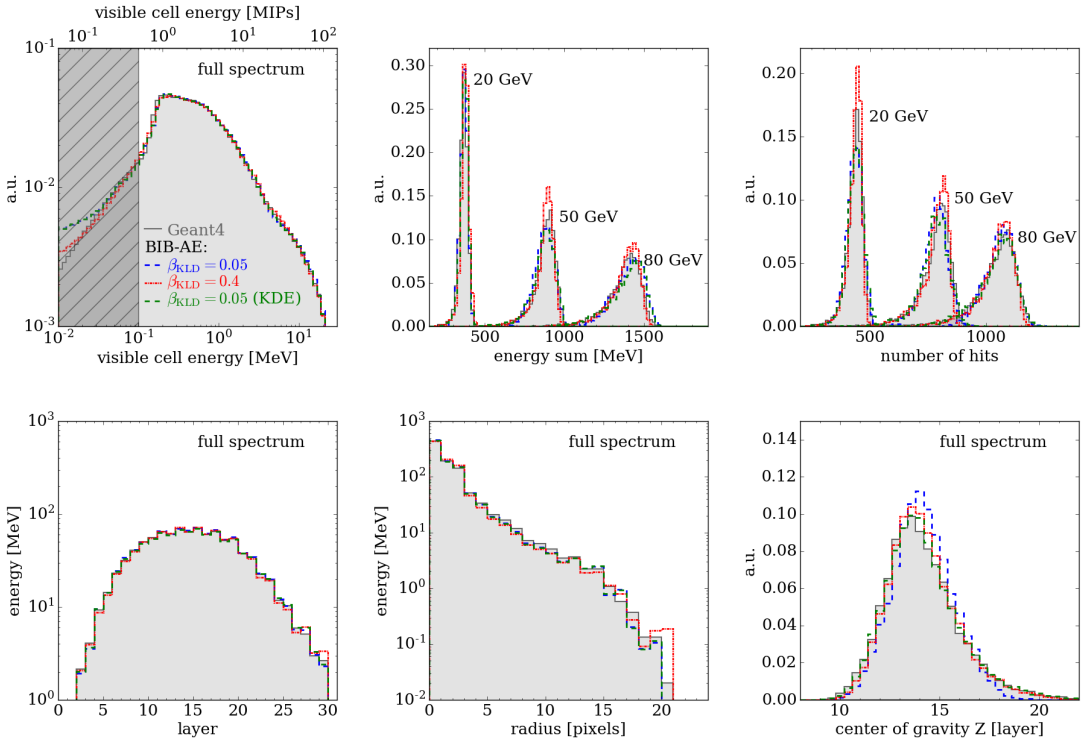


Figure 5.7: Differential distributions comparing various shower physics observables between GEANT4 and BIB-AE models with $\beta_{KLD} = 0.05$, $\beta_{KLD} = 0.4$ and $\beta_{KLD} = 0.05$ with KDE sampling. 40,000 showers are shown for the full energy spectrum and 2,000 showers for each single energy. Figures originally published in Reference [1].

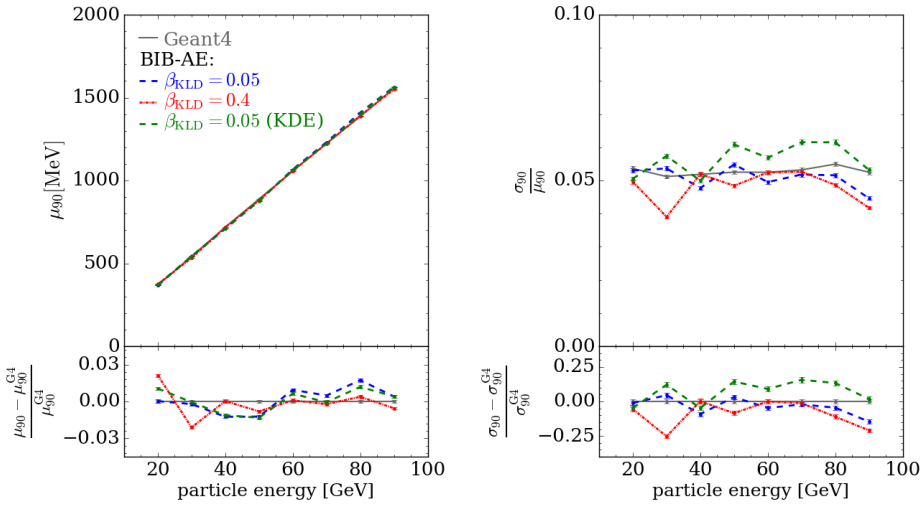


Figure 5.8: Mean and relative width (similar to the energy resolution) of the total visible energy deposited in the calorimeter for multiple incident particle energies for GEANT4 and BIB-AE models with $\beta_{\text{KLD}} = 0.05$, $\beta_{\text{KLD}} = 0.4$ and $\beta_{\text{KLD}} = 0.05$ with KDE sampling. Figure originally published in Reference [1].

Both models with $\beta_{\text{KLD}} = 0.05$ reproduce well the single energy distributions, however the model with $\beta_{\text{KLD}} = 0.4$ results in slightly too narrow distributions. We already observed in Reference [6] a slight mismodeling of the center of gravity in z distribution of the baseline BIB-AE model. This distribution is significantly improved with the newly trained model and with the KDE sampling, especially in the high center of gravity tail of the distribution. All three models perform equally well in reproducing the longitudinal and radial energy profile, however, for the $\beta_{\text{KLD}} = 0.4$ we observe a mismodeling at a high radius.

5.4.2 Linearity & Fidelity Score

To further benchmarks we show in Figure 5.8 the mean and relative width of the total visible energy distribution for multiple fixed incident energies between 20 and 90 GeV. Here the mean μ_{90} and width σ_{90} are calculated based on the 90 percentile of the distribution with the lowest root mean squared (RMS), as is customary for high-granular calorimeters made for particle-flow analyses [25]. These plots represent the linearity of the calorimeter response and calorimeter resolution (although it is not exactly the resolution as no calibration for the different sampling fractions is applied).

The linearity of all three models compared to GEANT4 is well reproduced. Outliers do not exceed more than 3%. Notably both the $\beta_{\text{KLD}} = 0.05$ and its KDE sampling variant perform very similarly. Further, all three models represent the relative width well, however, the KDE sampling model overestimates the width compared to the baseline, while the $\beta_{\text{KLD}} = 0.4$ model shows an outlier of almost -25% at 30 GeV. The baseline BIB-AE exhibits the best performance in this quantity.

Table 5.2: Fidelity score S_{JSD} for multiple model and sampling configurations of BIB-AE models with a latent size of $n = 24$. For $\beta_{\text{KLD}} = 0.05$ the best score out of multiple training runs is given – incidentally the model from Reference [6] – while the mean score and standard deviation for those trainings is: $\overline{S}_{\text{JSD},24} = 1.02 \pm 0.12$. For $\beta_{\text{KLD}} = 0.4$ only one training was performed. Tabel adapted from Reference [1].

config.	$\beta_{\text{KLD}} = 0.05$	$\beta_{\text{KLD}} = 0.4$	$\beta_{\text{KLD}} = 0.05$+KDE sampling
S_{JSD}	0.83	0.88	**0.67**

Overall, based on the six histograms and the single energy linearity and resolution, the baseline model with KDE sampling performs best, since it models all distributions similarly well as the baseline without KDE sampling, including the single energy distributions, and improves the center of gravity distribution. This is consistent with our earlier exploration of the latent space correlations, where we found the center of gravity to be encoded in the latent space.

To quantify the fidelity of the model variant further, we present their fidelity score S_{JSD} in Table 5.2. The fidelity score is calculated based on the differential distributions shown in Figure 5.7 and is lowest for the baseline model with KLD sampling ($S_{\text{JSD}} = 0.67$). This is explainable by performing in most distributions similar to the baseline BIB-AE ($S_{\text{JSD}} = 0.83$), better in the single energy distributions than the $\beta_{\text{KLD}} = 0.4$ model ($S_{\text{JSD}} = 0.88$), and much better in the center of gravity distribution than the baseline.

From these results, we conclude that between the two options of improving the latent space sampling with either (1) decreasing the information in the latent space by increasing the KLD weight β_{KLD} or (2) sampling the latent space from KDE of the encoding, the latter yields a BIB-AE model with superior generative fidelity.

5.4.3 Latent Kernel Density Estimate

We can further visualize this by investigating the differential distributions of the two highest KLD latent variables z_o and z_1 for both models with $n = 24$ and $\beta_{\text{KLD}} = 0.05$ and $\beta_{\text{KLD}} = 0.4$. The resulting distributions are shown in Figure 5.9 together with normal Gaussian distributed samples and samples drawn from a KDE of the $\beta_{\text{KLD}} = 0.05$ model $\boldsymbol{z}$ encoding. We see, that the $\beta_{\text{KLD}} = 0.05$ model latent values deviate from a normal distribution, but that the KDE can precisely replicate them. The model with $\beta_{\text{KLD}} = 0.4$ follows much more closely the normal Gaussian distribution, hence less information is encoded. Considering that the most information-rich variable z_0 encodes the center of gravity, sampling it from $\mathcal{N}(0, 1)$ yields a mismodeling, but the KDE preserves the information. Hence, we observe how the KDE sampling helps to preserve information encoded in the latent space resulting in an overall improved BIB-AE model.

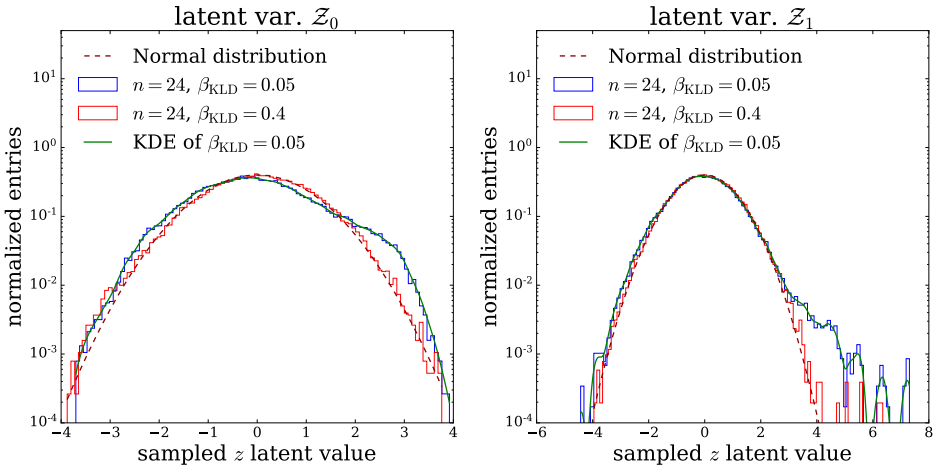
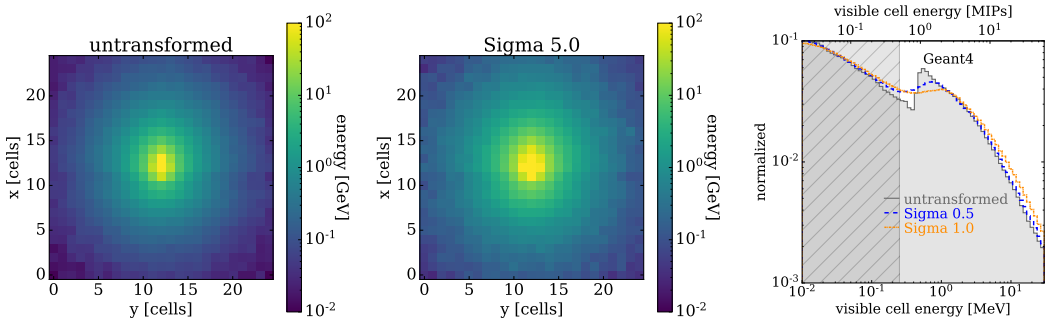


Figure 5.9: Sampled z_i values of the highest (left) and second-highest (right) KLD latent variables for 50,000 showers for BIB-AE models with a latent size of $n = 24$ and $\beta_{KLD} = 0.05$ or $\beta_{KLD} = 0.4$. Additional lines are added for a Normal distribution and the kernel density estimate (KDE) of the $\beta_{KLD} = 0.05$ model. Figures originally published in Reference [1].

5.5 Evaluation of Evaluation Scores

For any kind of generative modeling effort, it is important to decide on robust evaluation methods to compare the fidelity of the samples generated with generative models to the ground truth (in HEP often full simulations). Generally, the evaluation methods can be classified into qualitative and quantitative methods: Qualitative methods are based on the visual inspection of visualizations of the individual generated samples or differential distributions of low- or high-level observables calculated from a large number of generated samples. Therefore, such visual inspections are highly subjective. Quantitative methods are based on calculated scores that either compare individual generated samples to the ground truth or differential distributions of calculated observables. Such scores are often related to optimal transport, i.e. the Wasserstein distance or the maximum mean discrepancy (MMD). Quantitative scores can appear objective, however there are often still subjective choices involved such as the choice of observables used or how the scores are calculated. This is even true for the "classifier score" where a decision has to be made whether high- or low-level observables are used and what kind of classifier network is trained. Overall there is no such thing as the one "ultimate score", there are always trade-offs between them and the choice of score depends on the use case.

To understand this trade-off and to make more educated choices about which scores to use for the evaluation of calorimeter shower generative models, we investigate the Wasserstein-1 distance, the MMD, and the AUC of multiple low-level and high-level classifiers. We focus on evaluating the scores on their ability to distinguish between generative models, not whether a generative model is actually "good enough" to be used in a physics analysis since the latter question is highly dependent on the use case.



(a) Projection on z-axis **(b)** Smeared projection **(c)** Cell energy

Figure 5.10: **(a)** Overlay of 5,000 pion showers projected along the z-axis. **(b)** Overlay of 5,000 pions showers smeared with a $\mathcal{N}(1, 5^2)$ distribution projected along the z-axis. **(c)** Cell energy distribution of 40,000 showers of the pion dataset without transformation, with $\mathcal{N}(1, 0.5^2)$, and with $\mathcal{N}(1, 1)$ smearing. Figures taken from Reference [17].

5.5.1 Data Augmentation

To evaluate the scores, we use Gaussian-smeared pion showers with variable levels of smearing. As the unperturbed baseline dataset, we use the charged pion shower dataset from Reference [9] and apply various levels of perturbations to the data. The positively charged pions were simulated in the envisioned highly granular analog hadron calorimeter (AHCal) of the ILD. The AHCal uses 3×3 cm^2 scintillator tiles read out with silicon photomultipliers (SiPMs). A total of 48 active layers with stainless steel absorber plates make up the AHCal volume. The pions are simulated with a uniformly distributed incident energy between 10 and 100 GeV and a fixed angle and incident point. Showers in a subsection of the AHCal with the size of $25 \times 25 \times 48$ cells (pixels) are used as the 3D image dataset for this study — each 25 cells in transverse x and y direction and 48 cells in longitudinal z direction. For the simulation GEANT4 was used with the detector model implemented in DD4hep. A total of 510,000 pion showers were generated for this dataset — half of them are used for the perturbations and the other half is used as the baseline dataset.

As perturbation to the dataset, we smear the cell energy distribution by multiplying each cell energy with a random value $z \sim \mathcal{N}(1, \sigma^2)$. By varying σ and applying the half-MIP cut at 0.25 MeV, we create multiple transformed datasets for comparison with the baseline (untransformed) data. This Gaussian smearing is motivated by a common failure mode of calorimeter generative models which might not be able to precisely predict the cell energy, i.e. the GAN and WGAN from Reference [6] which do not model the "MIP-peak" well. Note that we also create a $\sigma = 0.0$ dataset, which is equivalent to the baseline dataset, but with different events. A comparison of the baseline and the smeared datasets is shown in Figure 5.10. By comparing the overlays of 5,000 showers projected along the z-axis in Figures 5.10a (untransformed) and 5.10b (transformed with $\sigma = 5.0$), we observe that the smearing essentially widens the shower core and increases the overall deposited energy as cell hits that were below the half-MIP cut are pushed above the threshold. For smaller

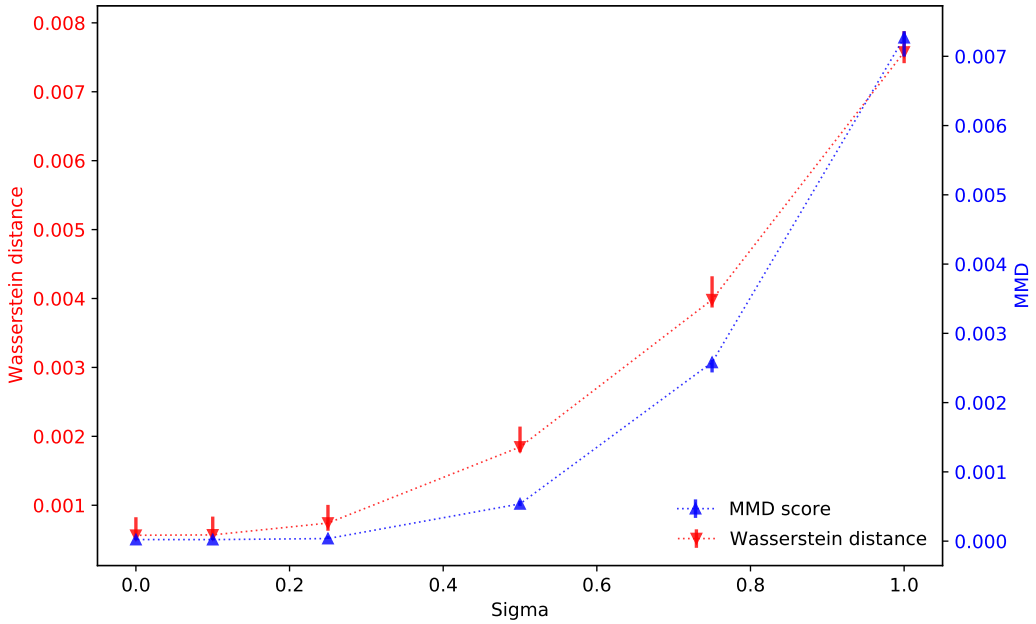


Figure 5.11: Wasserstein distance and energy-distance MMD between the baseline and Gaussian smeared pion datasets with increasing σ calculated based on 106 high-level observables. For each perturbation, the median score over five subsets of the baseline and perturbed datasets is shown, and the error bars indicate the minimum and maximum scores of the five subsets. Figure taken from Reference [17].

$\sigma = 0.5$ and $\sigma = 1.0$ this can also be seen in Figure 5.10c where the cell energy distribution is smeared out compared to the untransformed dataset and hence the MIP-peak is less pronounced. More data augmentations, i.e. adding a fixed single cell hit with variable energy deposition or adding a variable number of additional hits with a fixed energy, are investigated in Reference [17].

5.5.2 MMD and Wasserstein Distance Evaluation

We investigate how sensitive the Wasserstein-1 distance and a multi-dimensional energy-distance MMD [236] are to the Gaussian smearing applied to the pion dataset. These measures have the advantage that no hyperparameter choice is necessary. For the MMD, the energy distance kernel is given by $k(x, y) = -\|x - y\|$.

For reference, the MMD between two distributions P and Q is defined as [200]:

$$\mathrm{MMD}^2(P, Q) = \mathbb{E}_{(x,x')\sim P}[k(x, x')] + \mathbb{E}_{(y,y')\sim Q}[k(y, y')] - 2\mathbb{E}_{x\sim P, y\sim Q}[k(x, y)] \tag{5.8}$$

where x and x' are independent random samples from P (analogous for y and Q). In the energy distance MMD, the energy distance kernel $k(x, y) = -\|x - y\|$ is used. The Wasserstein-1 distance between two probability measures μ and ν is defined as [237]:

$$W_1(\mu, \nu) = \inf_{\gamma\in\Gamma(\mu,\nu)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{5.9}$$

where $\Gamma(\mu, \nu)$ is the set of all probability distributions with marginals μ and ν.

Both measures are used to evaluate the datasets based on high-level observables calculated from the pion showers. In total 106 features are calculated, including the total visible energy, the number of hits, the center of gravities in x-, y-, and z-direction (3 features), the first moment in $x-$, $y-$, and z-direction (3 features), and the energy profiles summed along the $x-$, $y-$, and z-axis ($25 + 25 + 48 = 98$ features). To stabilize the numerical variance of the features, they are normalized by the maximum values of the features from the baseline dataset. The Wasserstein-1 distance can be efficiently applied to one-dimensional distributions, while for multi-dimensional distributions it becomes costly to compute. Therefore, we calculate the Wasserstein distance between the baseline and each of the transformed datasets for each of the 106 features separately. For the total Wasserstein distance, we quote the mean over all 106 separately calculated ones. The energy-distance MMD on the other hand can be efficiently computed for multi-dimensional distributions. Hence, we can calculate the MMD between the baseline and each of the transformed datasets directly based on the 106-dimensional feature space. To estimate an error, we split the baseline and the transformed datasets into five equally sized subsets and calculated the scores for each of the five subsets separately (51,000 showers per subset). For the calculation of the Wasserstein-1 distance, we use the SciPy [238] implementation and for the energy-distance MMD, we use the GeomLoss library introduced with Reference [236].

The results are shown in Figure 5.11. We observe that both measures are similarly sensitive to the Gaussian perturbation, however at $\sigma \leq 0.25$ the MMD saturates while the Wasserstein distance only saturates at $\sigma \leq 0.10$. This means that at $\sigma = 0.25$ the Wasserstein distance is already sensitive to the perturbation, while the MMD is not. Above $\sigma = 0.25$, neither measure saturates and both behave similarly allowing us to distinguish between increasing levels of Gaussian smearing. This would allow us to use both measures to distinguish even stronger $\sigma > 1.0$ levels of distortion. Note that for this study the absolute scale of the measures is irrelevant, only their gradient is important for the comparison of differently modeled (perturbed) datasets. Overall it appears that the mean of the 1-dimensional Wasserstein distances is more sensitive to small dataset perturbations than the multi-dimensional energy-distance MMD. This might be because for very similar datasets the MMD is subject to vanishing gradients due to electrostatic screening [239]. Additional studies with the MMD and Wasserstein measures using further perturbation on the pion dataset are presented in Reference [17].

5.5.3 Classifier Evaluation

As a "classifier score" we use the AUC score of a classifier trained to distinguish between the baseline and the perturbed datasets. We contrast four different classifier setups: a low-level classifier with a fully connected architecture (15M parameters), a low-level classifier with a convolutional architecture (1.4M parameters), a high-level classifier with a fully connected architecture using all 106 features (22k parameters), and a high-level classifier with a fully connected architecture using 8 features (all observables mentioned above except the 98 energy profiles) (9k parameters). The low-level classifiers are trained on the 3D images directly, while

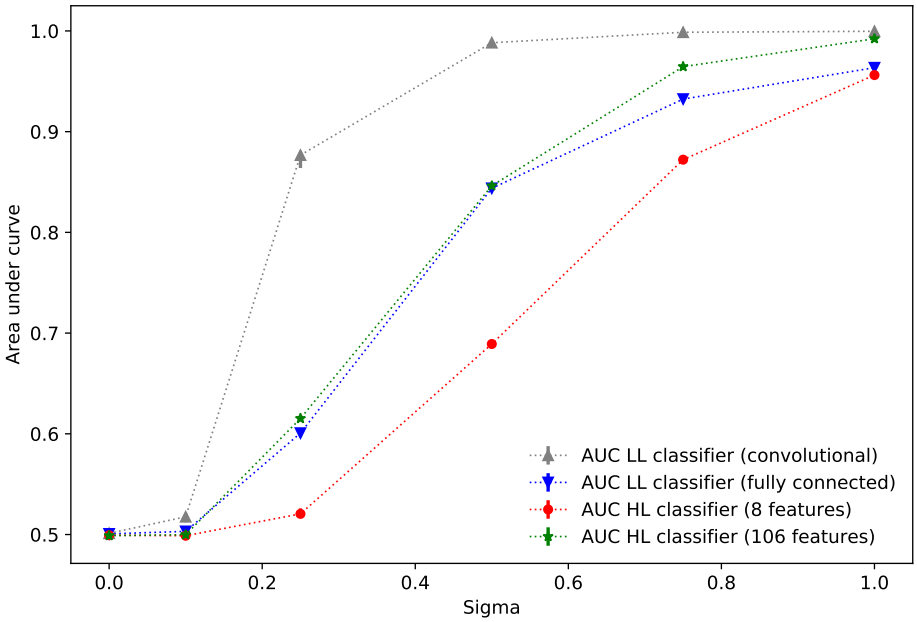


Figure 5.12: AUC scores of four different classifier setups trained to distinguish between the baseline and Gaussian smeared pion datasets with increasing σ. For each perturbation, five trainings with different datasets splits are performed. The median score is shown as well as the minimum and maximum scores as error bars. Figure taken from Reference [17].

the high-level classifiers are trained on the calculated observables. The CNN architecture consists of two 3D convolution layers and four fully connected layers. All three fully connected classifier consists of three dense layers. As activation functions, LeakyReLU [230] ($\alpha = 0.1$) is used for all layers except the last one, where a sigmoid activation is used. All classifiers are implemented with PyTorch [173]. More details on the architectures and the training are given in Reference [17].

For each classifier setup, we train five times with different dataset splits, i.e. 77% training, 8% validation, and 15% test set. For each split different events are in the test set and in each set 50% are from the baseline and 50% from a perturbed dataset. Hence the test set for evaluation contains 76,500 showers. This way we can show how consistent the classifiers are trained and how stable the AUC score is, since it is a requirement for any useful evaluation score for generative models to allow for a high degree of reproducibility [240].

The median AUC scores of the five classifiers for various levels of Gaussian smearing with $\sigma \in [0, 1]$ are shown in Figure 5.12 together with the minimum and maximum scores as error bars. Overall, all classifiers approach a high AUC score of ≥ 0.95 at $\sigma = 1.0$. We observe that at $\sigma = 0.1$ the low-level convolutional classifier has a distinctly higher AUC > 0.5, while the other classifiers cannot distinguish between the baseline and the perturbed dataset. Above $\sigma = 0.5$ the low-level convolutional classifier saturates at an AUC ≈ 1.0 and therefore the AUC score of this classifier cannot be used anymore to distinguish between levels of perturbation. The other classifiers appear to saturate when approaching $\sigma = 1.0$, but still exhibit a strong gradient between $\sigma = 0.5$ and $\sigma = 1.0$. Interestingly, both the

low-level fully connected and the high-level 106 features classifier behave very similarly. The high-level 8 features classifier picks up and saturates the slowest, making it possibly the best choice for distinguishing between even stronger perturbations $\sigma > 1.0$. For this purpose, the low-level convolutional classifier may not be the best option, as it tends to reach saturation the fastest. Yet if a generative model produces events already at a very high fidelity, we would expect rather small perturbations and here the low-level convolutional classifier might be a good choice as it is the most sensitive to small perturbations. Overall both very powerful (i.e. low-level convolutional classifiers) and less accurate classifiers (i.e. high-level classifiers with limited features) have their raison d'être, as depending on the fidelity of a generative model one or the other is best suited to evaluate which is best. This conclusion can also be drawn from studying these classifiers with other perturbations in Reference [17].

A notable difference between using either MMD and Wasserstein distances or classifiers for model evaluation is the evaluation time: While the training of a classifier takes several hours on a GPU, the calculation of the Wasserstein distance and the MMD takes only a few minutes on a CPU. Especially for small-scale studies, this makes the Wasserstein distance and the MMD more attractive evaluation metrics than the classifier AUC score. When it comes to the related question of whether a generative model faithfully reproduces the ground truth, a powerful classifier is likely a very good choice as it can distinguish between even small perturbations in a multivariant way. However, this specific question was not investigated in this study. More details on classifiers as evaluation tools for generative models can be found in Reference [241].

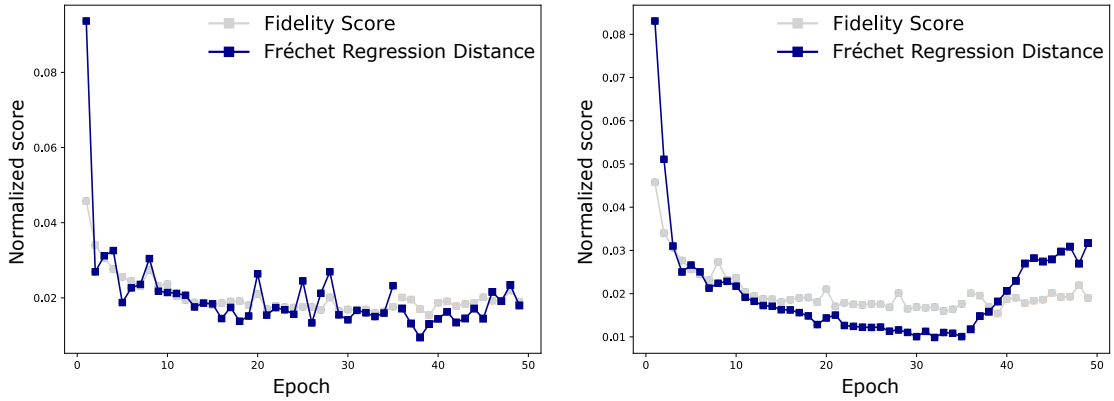
5.6 Fréchet Regression Distance

In computer vision, a popular evaluation method for generative models is the Fréchet Inception Distance (FID) [242]. It is calculated as the Fréchet distance between a (usually low-dimensional) embedding of the generated samples and the embedding of the real samples. As embedding, the FID utilizes the activations of the second-to-last layer of the Inception network [243], a multi-classification network trained on the ImageNet dataset [244]. It was adapted for the evaluation of generative models for jets as particle clouds with the Fréchet ParticleNet distance (FPD) [62], where the Inception network is replaced with a multi-classifier based on the ParticleNet architecture [245].

As part of an ongoing effort to investigate novel evaluation methods for calorimeter shower generative models, we explore a custom implementation of the FID based on a regression network for calorimeter shower energy reconstruction. We dub this score the *Fréchet regression distance* (FRD). Previous custom implementations of the FID have only explored classifiers as embedding networks, but since the Fréchet distance is calculated based on the activations of the embedding network, a regression model is technically also a valid choice.

In this section, we outline our exploration of the FRD based on a simple regression neural network. Further studies with more complex regression networks as well as the application of the FID itself to calorimeter shower images can be found in Reference [16]. We present here the most representative results of this study.

The regression network is the same as the energy constrainer network used as part of the



(a) Trained network **(b)** Untrained network

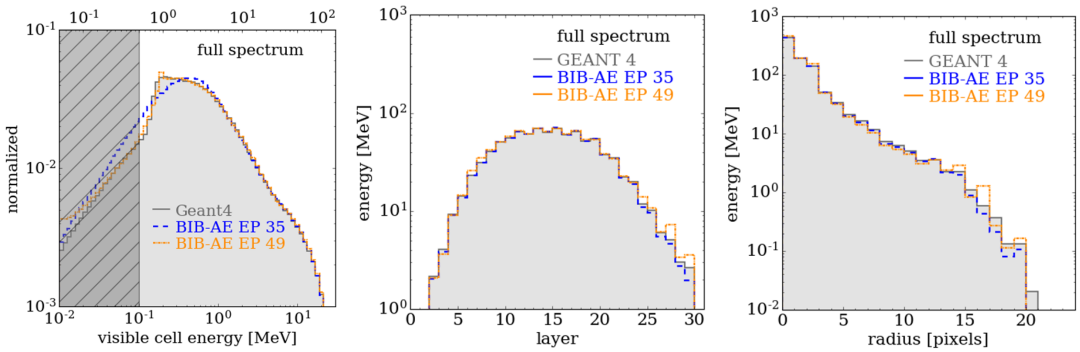
Figure 5.13: Fréchet regression distance between GEANT4 and BIB-AE generated showers for various training epochs using the trained regression network **(a)** and a randomly initialized model **(b)**. For comparison, the fidelity score S_{JSD} is shown as well. Note that the BIB-AE training up to epoch 35 does not include the Post Processor Network, after epoch 35 it is enabled. Figures adapted from Reference [16].

WGAN model in Reference [6]. It consists of three 3D convolutional layers intertwined with layer normalization followed by two fully connected layers. All layers use LeakyReLU as the activation function, except the last layer which uses ReLU. The second to last layer has 100 activations, which are used for the calculation of the FRD.

For studying the FRD, we use the ILD ECAL photon shower dataset in 3D calorimeter image format as introduced in Section 5.1. The regression network is trained on a subset of the data using 80,000 showers with a uniform energy distribution between 10 and 100 GeV. The validation set consists of 20,000 showers. We train the network for 200 epochs using a mean absolute error (MAE) loss with the default ADAM optimizer and choose the model with the lowest validation loss as the final model.

To evaluate the effectiveness of the FRD, we apply for the comparison of the GEANT4 test set with various BIB-AE generated datasets. For the generated showers we use the epoch-wise checkpoints of the BIB-AE training performed for Reference [6]. The GEANT4 test set includes 49,800 showers and the same amount is generated with the BIB-AE model for each epoch of the BIB-AE training up to epoch 49. Note that for this training between epochs 1 and 35, the Post Processor Network (PPN) is not used, while after epoch 35 it is enabled. For the results in Reference [6], epoch 39 was chosen as the best epoch. This is also the epoch with the lowest fidelity score S_{JSD}. The fidelity score is a combination of JSDs calculated from multiple calorimeter shower observables and is therefore physically motivated (see Section 5.3.1). It is a good baseline for comparing the FRD to, since naively we would expect them to behave similarly.

For all epochs of the BIB-AE training, the FRD and the S_{JSD} are shown in Figure 5.13a. We observe that compared to the S_{JSD}, the FRD is more sensitive to the training progress of the BIB-AE model and varies stronger between epochs. We observe the lowest FRD for epoch



(a) Cell energy **(b)** Layer energy **(c)** Radial energy

Figure 5.14: Cell energy distribution **(a)**, longitudinal energy profile **(b)**, and radial energy profile **(c)** comparing photon showers simulated with GEANT4 and BIB-AE generated showers with 35 (without post-processing) and 49 (with post-processing) epochs of training. 40,000 showers are shown for the full energy spectrum and 2,000 showers for each single energy. Figures taken from Reference [16].

38, but the previously chosen best epoch 39 is among the ones with the lowest FRD. Similarly to the S_{JSD}, the FRD increases again towards the end of the post-processing training after epoch 40. Interestingly some high FRD outliers correlate with high S_{JSD} outliers, but not all. Unlike the fidelity score, which offers physical interpretability via the underlying differential distributions, it is not clear what the FRD is sensitive to. Naively we would expect it to be sensitive to showers that allow energy reconstruction similar to the training data. We also investigate a few deeper regression networks, but the progression of the resulting FRDs proves to be inconclusive. Detailed results can be found in Reference [16].

To further study the FRD, we calculate it also using the regression network in the untrained state, i.e. with randomly initialized weights. The resulting FRD is shown in Figure 5.13b. We observe that the untrained FRD is more stable than its trained counterpart and decreases almost monotonic with the BIB-AE training epochs until epoch 35. With the post-processing enabled, the FRD increases again until epoch 49 where its value is similar to the one at epoch 3. Hence, according to the untrained FRD, serious mismodeling is introduced with the combined BIB-AE and PPN training.

To understand this behavior, we compare the cell energy distribution, the longitudinal energy profile, and the radial energy profile of the GEANT4 test set with the BIB-AE generated showers for epochs 35 and 49 in Figure 5.14. As expected, the cell energy distribution — in particular the MIP peak — is better modeled with the post-processing enabled, however, the longitudinal and radial energy profiles are better modeled without the PPN. Specifically, the generated showers by epoch 49 exhibit outliers in the later layers of the calorimeter and at high radii, indicating that too much energy is deposited at the outer edges of the calorimeter. This likely results in showers that are separable from the GEANT4 test set in the embedding space of the untrained regression network resulting in the high FRD. However, such outliers would likely not be reconstructed and might therefore in practice not be of high relevancy for a physics analysis.

In conclusion, the FRD is an interesting evaluation measure for calorimeter generative models but is held back by its lack of interpretability. When applied with a simple regression model, the metric overall seems to pick up on some physically important changes in the modeling, however, it is not clear which changes exactly. Using the FRD with a randomly initialized model might be a promising research direction since the random embedding might more generally pick up on dataset features that are not necessarily important for the regression training. However, these features might also not be physical at all or could be ignored for a downstream task. Considering portrait photos of people, an analogy for this behavior is that a model is rather sensitive to the background, however as an observer, one might only be interested in the person. It appears that when untrained, the FRD is sensitive to any kind of change in the picture, regardless of physically important or not. For further research, more physical observables could be predicted by the regression network which might overall improve the reliability of the FRD as a generative modeling evaluation measure.

5.7 Summary

Generative models are a promising tool to speed up the simulation of particle showers in calorimeters, which are traditionally simulated with Monte Carlo-based simulators such as GEANT4 and therefore require a lot of computational resources. To apply machine learning models originally developed for computer vision tasks, the energy depositions in calorimeter cells are represented as 3D images. This allows us to deploy convolutional neural networks for the modeling of calorimeter showers. In this chapter, we have discussed the application of the BIB-AE model to the generation of photon calorimeter showers. Since large models such as the BIB-AE require resource-intensive hyperparameter optimizations and training efforts, we study how a better understanding of the encoded latent space can increase the model's generative fidelity.

We first investigate the size of the latent space and find that for a fixed $\beta_{\text{KLD}} = 0.05$ the information content encoded saturates around 45 nats. However, we find the best-performing model – measured by the physically motivated fidelity score S_{JSD} – at $n = 24$ and an information content of about 29 nats. This indicates that more encoded information does not necessarily lead to a better generative model meaning that some information that is contained in the latent space does not contribute to the generative fidelity. An interesting open research question is, therefore, how the "useful" information in the latent space could be quantified.

Most information of the latent space is consistently encoded in the first two latent variables. To understand what kind of physical information might be stored in the latent space, we study the correlations between the five most information-rich latent variables and several shower observables. We find that consistently across model configurations, the first latent variable encodes the center of gravity (first moment) in z-direction, while the second latent variable encodes the second moment in z-direction. This kind of encoding can be leveraged to generate calorimeter showers with for example a specific center of gravity or shower start layer. Further, more precise modeling of the encoded latent space can improve the overall generative fidelity of the BIB-AE. This can be achieved with two options: (1) decreasing the information

content in the latent space by increasing the KLD weight β_{KLD} in the total training loss and (2) sampling the latent space from a KDE of the encoding. Pushing the latent distributions more towards a unit normal distribution enhances the modeling of physical observables that have the strongest correlation with the most information-rich latent variables , i.e. the center of gravity in z. However, this results in a decrease in the performance of the other variables such as the number of hits. The latter strategy proves to be the most effective, providing the added advantage of being easily applicable to the previously trained BIB-AE model (or any other VAE-like model).

With the increasing applications of generative models in particle physics, a deeper understanding of the model's inner workings helps establish trust in their generative fidelity and can help to improve them. Considering the abundant amount of labeled simulated data in particle physics, these datasets are uniquely qualified for comprehensive studies of generative models and their latent spaces. The physics understanding of the data offers the opportunity for detailed model studies that can improve the generative models and answer interesting questions about the balance of encoded information and generative fidelity.

To answer such questions various models need to be evaluated and their generative fidelity compared. This requires robust evaluation methods that can be applied to generative models for physics data such as calorimeter showers. We studied several previously proposed evaluation methods including the Wasserstein-1 distance, the MMD, and the AUC score of classifiers. Additionally, we introduced a custom implementation of the Fréchet inception distance (FID) based on a regression network, which we dub the Fréchet regression distance (FRD).

We find that the mean of one-dimensional Wasserstein-1 distances and the multi-dimensional MMD are similarly sensitive to a large range of Gaussian perturbations of the pion shower dataset. However, for small perturbation levels, the MMD saturates while the Wasserstein-1 distance is still able to distinguish between such small perturbations. The AUC of a classifier is another good evaluation measure but depends strongly on the choice of classifier architecture and the choice of observables used. Interestingly, we found that a less powerful classifier architecture might be more suited to distinguish between a large set of generative models. This is because a powerful classifier might already saturate (separate perfectly) at small perturbation levels, while a less powerful classifier is still able to distinguish between stronger perturbations. A further consideration is the evaluation time: Measures such as the Wasserstein distance and the MMD can be calculated in a few minutes on a CPU, while the training of a classifier takes several hours on a GPU.

The FRD offers an alternative way of using a neural network for the multi-variant evaluation of generative models. Since the FRD is based on a single regression model, it can be used as an evaluation measure as fast as the MMD and Wasserstein distance. However, it is currently held back by its lack of interpretability. This could be remedied by using explainability methods such as layer-wise relevance propagation (LRP) [246, 247] to understand which features the regression network is sensitive to. The use of an untrained network for the embedding might be another promising research direction.

Chapter 6

Point Cloud Modeling of Calorimeter Showers

The results presented in this chapter have previously been published in References [3, 5] in collaboration with Sascha Diefenbacher, Engin Eren, Frank Gaede, Gregor Kasieczka, William Korcari, Anatolii Korol, Katja Krüger, and Peter McKeown. The figures and tables are similar or identical to the ones published. My contributions in obtaining these results include the implementation of the EPiC Encoder and Latent Flow networks and the total loss objective in the CaloClouds models, the implementation and training of the CaloClouds II model, as well as the implementation of the consistency distillation for CaloClouds II (CM). I wrote parts of Reference [3], large parts of Reference [5], and addressed most of the reviewer comments during the publication processes. The discussion in this chapter closely follows the above-mentioned publications.

As seen in the previous Chapter 5, photon calorimeter showers can be well modeled as 3D images with a CNN-based generative model such as the BIB-AE. Other generative models for calorimeter shower generations include generative adversarial networks [39–48], autoencoder-variants [1, 6, 9, 49–51], normalizing flows [52–58] and diffusion models [59–61]. A GAN-based generative model for simulating the calorimeter response has already been successfully deployed by the ATLAS collaboration [154].

However, the photon shower images are very sparse, i.e. the number of image pixels is 27,000 while the average number of hits is around 1,000 — only around 4% filled pixels and the remaining are zero-padded. It is much more computationally efficient, to only generate these few percent filled pixels with a generative model. This can be done by representing the calorimeter shower as a point cloud $X = \{\boldsymbol{x}_i\}_{i=1}^{N}$ instead of a 3D image, where each point $\boldsymbol{x}_i$ is an energy deposition in the calorimeter with its 3D position and energy as features (4 features in total) and N is the cardinality.

A second advantage of calorimeter point clouds is, that more granular information than just cell hits can be used. The points can be clustered Geant4 steps, i.e. energy depositions on an ultra-high granular grid not accessible in experiments. This increases the number of points to be simulated but allows for a largely cell geometry-independent shower simulation. Within the calorimeter, the shower becomes translation invariant and can be projected anywhere

in the calorimeter (without changing its depth and the layer layout). These projections are less likely to produce artifacts due to cell gaps and staggering than if they were performed with regular cell-level granularity. Representing such ultra-high granular calorimeter showers as 3D images would be even more computationally inefficient than images with cell-level information. Hence, this geometry independence can currently only be achieved with the point cloud representation.

An early attempt at a point cloud generative model for calorimeter simulation using ordinary normalizing flows was presented in Reference [248], yet it still resulted in low-fidelity showers. After the publication of Reference [3], a comparison between an image- and point cloud-based diffusion model for pion showers was presented in Ref. [249], but it only considered low-granularity showers with up to 200 points per shower. Also, after the publication of Reference [3], *geometry-aware models* [250] were introduced as a complimentary way of achieving geometry independence by training an image-based autoregressive model on a dataset containing various calorimeter geometries.

This chapter focuses mainly on the implementation, the generative fidelity, and the computational efficiency of the CaloClouds and CaloClouds II models. Studies on the geometry-independence can be found in Reference [3].

The dataset and the point cloud representation of calorimeter showers are introduced in Section 6.1. The model components of the CaloClouds model are explained in Section 6.2 and the CaloClouds II as well as its distilled variant are explained in Section 6.3. We compare the performance of all three models against Geant4 with both qualitative and quantitative evaluation methods in Section 6.4. A chapter summary is provided in Section 6.5.

6.1 Calorimeter Showers as Point Clouds

The point cloud dataset[1] was created from simulated photons that shower in the electromagnetic calorimeter (ECAL) of the proposed International Large Detector (ILD). The ECAL is a highly granular sampling calorimeter made up of 30 active silicon sensor layers and passive tungsten absorber layers. The silicon sensor cells have a size of 5×5 mm with a thickness of 0.525 mm. Out of these 30 tungsten layers, the first 20 have a thickness of 2.1 mm and the remaining 10 have a thickness of 4.2 mm.

The showers are simulated using Geant4 Version 10.4 [118] (using the QGSP-BERT physics list) with the ILD detector implemented in the iLCSoft [227] framework. The ILD ECAL model is implemented in DD4hep [117]. It is a realistic model including air gaps between silicon sensors and position-dependent cell staggering, resulting in an irregular 3D grid.

To describe the origin of the simulated particles in the ILD, we use a global coordinate system denoted as $[X', Y', Z']$, where X' lays in the horizontal plane, Y' points vertically, and Z' parallel to the beam pipe. Simulated photons are produced at $[X' = 0, Y' = 1811.3 \text{ mm}, Z' = 40 \text{ mm}]$ and their trajectory points along Y'. As this position is right in front of the ECAL, interactions outside the ECAL are mitigated. This position was further

[1]The dataset simulation and processing was implemented by Anatolii Korol.

Table 6.1: Overview of the three types of point clouds, either on GEANT4 step-level, on clustered step-level ("points"), or on cell-level ("hits"). The number of points per shower (second column) indicates the maximum at 90 GeV. Table adapted from Reference [3].

	points / shower	Note
All GEANT4 steps	40 000	Initial output of GEANT4
Clustered GEANT4 steps	6 000	Input/output of CALOCLOUDS
Hits in calorimeter grid	1 500	Calculation of physics observables

chosen to avoid hitting large gaps in the ECAL itself. The incident photons are simulated with a uniform energy distribution between 10 and 90 GeV.

6.1.1 Clustering

In the full GEANT4 simulation, a large number of energy depositions, called GEANT4 *steps*, are created in the sensitive sensor material due to secondary particles traversing the calorimeter cells. For the simulated photon showers in the ILD ECAL, this results in up to 40,000 GEANT4 steps per shower. Usually, all GEANT4 steps in the area of one calorimeter cell are summed up and a resulting calorimeter *hit* with the summed up energy depositions is stored. These calorimeter hits can be directly compared to hits measured in real experiments, as the step information is only a byproduct of the simulation and is not available in reality.

Ideally, to save computing time for the full GEANT4 simulation, a generative model should create hits at the cell level. This is also what other generative machine learning models for fast calorimeter simulations do, i.e. the BIB-AE discussed in Chapter 5. Yet, the generation of discrete cell hits as a point cloud is difficult as small mismodelings like overlapping points can heavily impact the quality of the generated data in various observables, for example by changing the total number of hits N_{hits}.

Instead, one could train a generative model that emulates simulated GEANT4 steps. This also adds the above-discussed advantage of making the generative model cell-geometry independent, allowing for a projection of the shower anywhere in the calorimeter without adding reconstruction artifacts. However, this would result in a much more granular point cloud with up to 40,000 steps per cloud (at 90 GeV), which would be prohibitively expensive and difficult to compute.

Therefore, we introduce here a middle ground between all GEANT4 steps and simple cell hits. We cluster the GEANT4 steps within an ultra-high granular grid with 36× higher granularity than the real simulated cell sizes, using a square grid with sizes of 0.83×0.83 mm (and as thickness the cell thickness of 0.525 mm). All GEANT4 steps within one ultra-high granular cell are summed up and we denote these as *clustered steps*. The size of the ultra-high granular grid was tuned such that this results in a point cloud with up to 6,000 clustered steps at 90 GeV — a size which is computationally feasible to allow for a fast generative model, yet keeps the advantages of the full GEANT4 steps. An overview of the different point cloud sizes is given in Table 6.1.

6.1.2 Data Processing

We define a second local coordinate system $[X, Y, Z]$ near the impact point of the photon showers into the ECAL. Here, X and Y points parallel to the calorimeter layers, while Z is directed perpendicular to the layers along the trajectory of the incident particle. Since no GEANT4 steps are recorded outside the silicon layers, this results in discrete Z positions with various gap sizes. For a generative model, continuous distributions are easier to learn than discrete ones, therefore, these gaps in the Z direction are removed. The Z coordinate is smeared within the size of one layer, to achieve a continuous Z distribution between layers 1 and 30.

Since generative models work well with normalized inputs, a square box around the incident point is defined as $X_{\text{min}} = -200$ mm and $X_{\text{max}} = 200$ mm as well as $Y_{\text{min}} = -160$ mm and $Y_{\text{max}} = 240$ mm. All points outside are discarded (less than 4 ‰). As a final pre-processing step the X, Y, and Z features are normalized to a scale of [-1,1] based on their respective maximum value.

As part of the post-processing pipeline, the normalization is reversed and the layer positions are set to the exact positions in the layer centers of the simulated ECAL between $Z \in [1811.5 \text{ mm}, 2010.3 \text{ mm}]$. A visualization of the pre-processing pipeline is shown in Figure 6.1.

6.1.3 Simulated Statistics and Evaluation

For the training set, 525,000 photon showers with uniformly distributed incident energies between 10 and 90 GeV are simulated using GEANT4. For the evaluations in Section 6.4, multiple test sets are simulated: 40,000 showers uniformly distributed showers are used for the full spectrum evaluation plots; for the single energy evaluation plots at 10, 50, and 90 GeV, 2,000 showers for each energy are generated; and for the evaluation scores another 500,000 uniformly distributed showers are simulated.

For experimental applications and downstream analyses, only the cell-level energy depositions are available, hence, all comparisons between the GEANT4 simulation and generative models need to be performed on observables calculated from cell hits. All observables and evaluation metrics shown in this chapter are therefore derived from cell-level hits. For this purpose, the generated point clouds as well as the GEANT4 steps are binned using the real geometry of the ECAL such that for each event 30 layerwise 2D histograms with 30×30 bins are created.

6.2 CaloClouds Model

Point cloud generative models transform points sampled from random noise into points with a meaningful structure. If one were to order the points in a fixed size point cloud, i.e. an ordered list of features, this generative task would be comparatively easy as simple network structures such as fully connected layers can be used. However, a point cloud is usually defined as an unordered list of points with a variable cardinality. This directly reflects the sets of measurements taken at particle detectors, which are often unordered. To make a learning

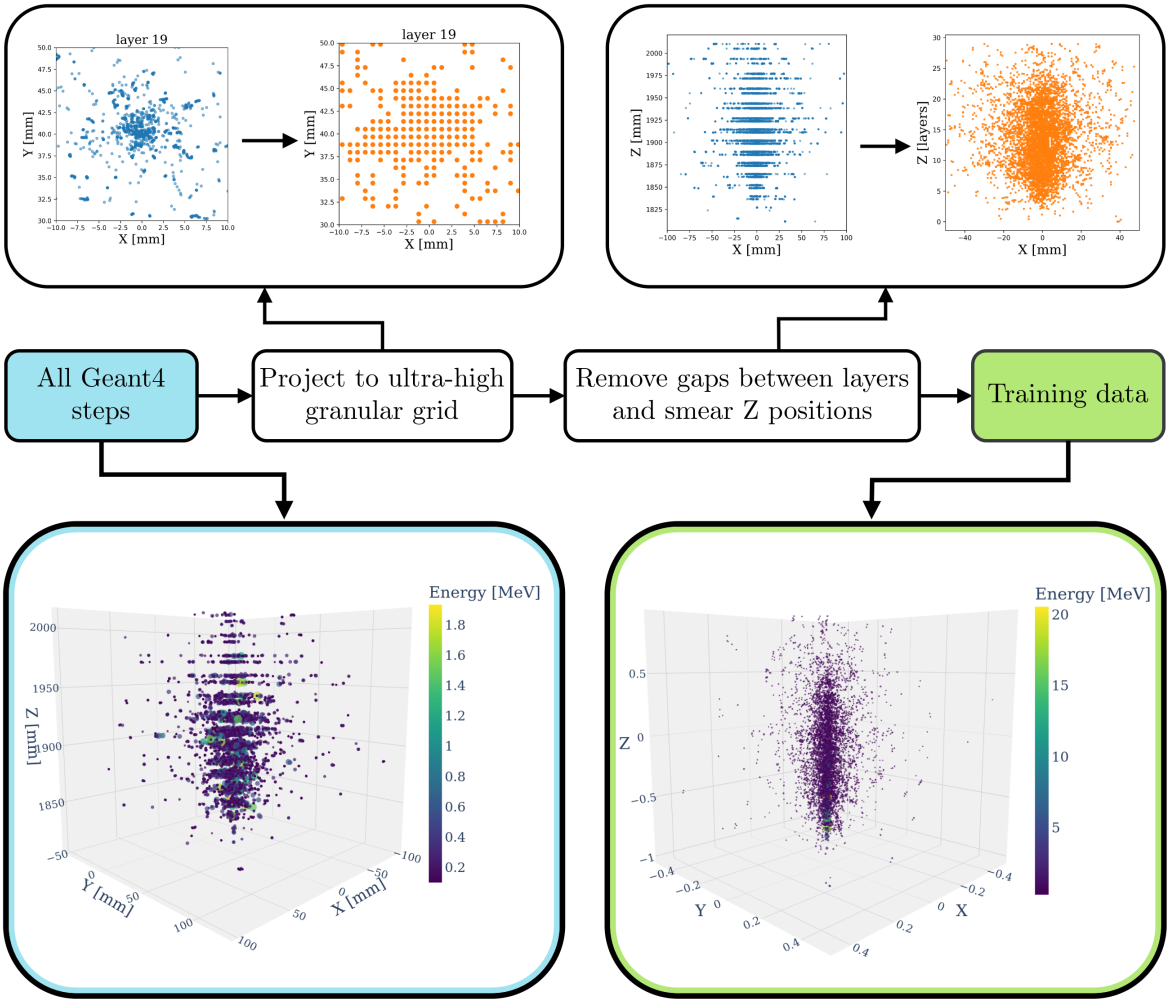


Figure 6.1: Overview of the data processing pipeline for the calorimeter showers used in CaloClouds. First, all Geant4 steps (up to 40,000) are clustered in an ultra-high granular grid (up to 6,000 points). Afterward, the gaps (air gaps and absorber material) between layers are removed and the z positions are smeared within its layer to achieve a continuous distribution. Figure originally published in Reference [3].

task on such point cloud data easier, the model architecture should reflect these attributes, i.e. functions learned should be permutation equivariant and applicable to a variable number of points. Of course, one could simply order the point cloud and apply zero padding to be able to use simple fully connected layers, but such a fixed neural network needs to learn to not use any unphysical ordering within the list of particles. A model with the inductive bias of permutation equivariance can therefore make the learning task easier and it ensures that indeed the output of the model is invariant to the order of the input points.

For generating calorimeter showers as point clouds, we introduce CaloClouds, a combination of two regular fixed-size neural networks, a permutation-invariant network, and a permutation-equivariant network. The two fixed-size neural networks are the *Shower Flow*, a normalizing flow responsible for generating conditioning variables and calibration factors, and the *Latent Flow*, another normalizing flow that generates a shared latent space used as conditioning for all points. During training, this latent space is created by a permutation-invariant *EPiC Encoder*, during sampling it is derived from the Latent Flow. The latent space is used as a conditioning for the permutation-equivariant *PointWise Net*, a diffusion model that transforms randomly sampled points into a physically meaningful calorimeter shower representation. Core parts of this model architecture are inspired by Reference [251]. Additionally, the CaloClouds model involves several calibration steps to achieve a high-fidelity calorimeter shower generation. In the following these different generative models and their purpose during training and sampling are explained. For a detailed discussion of the generative modeling paradigms used, see Chapter 4. An overview of the training and sampling pipelines of CaloClouds are shown in Figure 6.2.

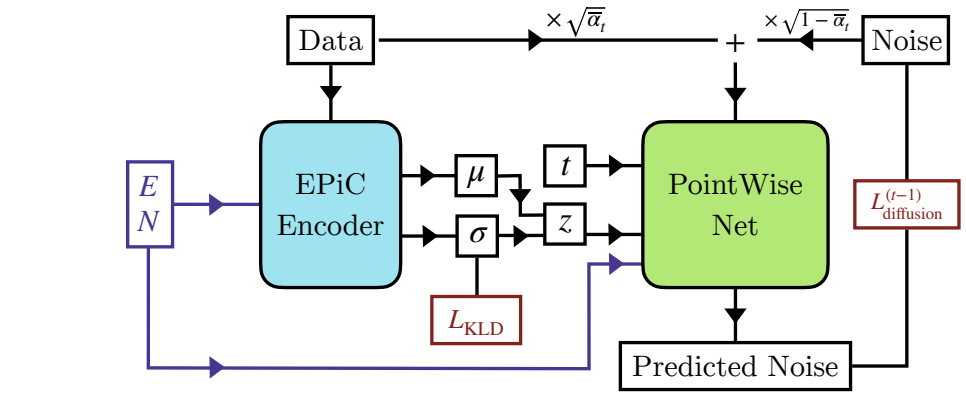
6.2.1 PointWise Net

Core to the CaloClouds model is the diffusion model dubbed *PointWise Net*[2]. In CaloClouds, we use the denoising diffusion probabilistic model (DDPM) [214] paradigm (see Section 4.4.1). The DDPM is implemented with $T = 100$ time steps (for training and sampling) and uses a quadratic variance scheduler with $\beta_T = 0.02$ and $\beta_1 = 10^{-4}$. The architecture used is based on fully connected layers with weight sharing across all points. This way the model is applied to every single point individually and therefore permutation equivariant. PointWise Net is a conditional model conditioned on the incident energy E, a latent space $\boldsymbol{z}$ shared across all points, and the number of points N of the point cloud — also used to sample the appropriate amount of random points to be transformed (or "denoised"). As for all diffusion models, the time step t is an additional conditioning feature.

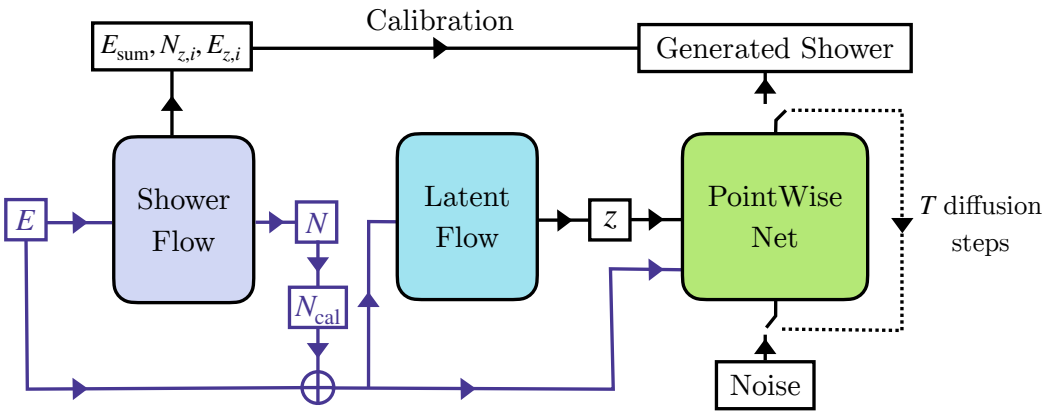
The PointWise Net itself consists of five *ConcatSquash layers* (CSL), inspired by Reference [252]. The exact architecture and layer structure are shown in Figure 6.3. The input and output dimensionalities of the model are four features: three positional point features $x_{\text{point}}, y_{\text{point}}$, and z_{point} and the point energy E_{point}. Every CSL is individually conditioned on the context features. The CLS can be viewed as a realization of the equivariant Deep Sets concept [176] (see Section 3.5).

Using the global conditioning including the learned latent vector $\boldsymbol{z}$, every photon shower

[2]Model taken from Reference [251] and adapted by Engin Eren and Anatolii Korol.

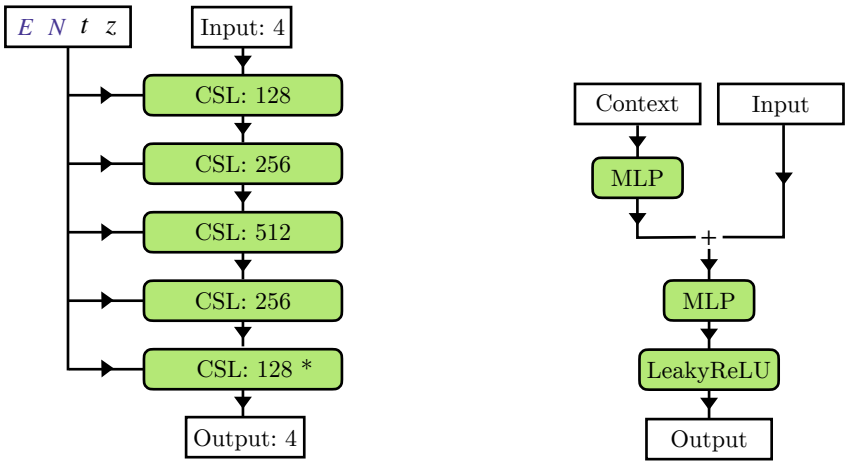


(a) Training at random time step t



(b) Sampling with reverse diffusion through all time steps T

Figure 6.2: Overview of the training and sampling pipeline of the CaloClouds model. The separate training of the shower and latent flow are not shown. Figures adapted from Reference [3].



(a) PointWise Net **(b)** ConcatSquash layer (CSL)

Figure 6.3: (a) Architecture of the PointWise Net in the CaloClouds model with the number of dimensions indicated. **(b)** Layout of a ConcatSquash layer (CLS) containing multiple multi-layer perceptrons (MLP). *In the last layer, no activation function is applied. Figures originally published in Reference [3].

point is sampled independently and identically distributed (i.i.d.). The i.i.d. assumption simplifies reality as photon showers are a cascade of secondary particles that develop longitudinally in time through the calorimeter. However, for photon showers this results in a simple topological shower structure, and even for complicated point cloud structures, i.e. a point cloud representing a plane or a chair, this assumption appears to work well as shown in Reference [251]. Future developments of point cloud generative models for more complex hadronic calorimeter showers likely will need to implement models that account for inter-point correlations. This could be done with more complex model architecture such as self-attention transformer networks [180], graph networks [177], sequence convolutions [253], or equivariant point cloud (EPiC) layers (see Chapter 7). A disadvantage of these more complex layers is their computational cost. These models are slower than the PointWise Net and for $\mathcal{O}(1000)$ points and diffusion models with $\mathcal{O}(10-100)$ steps this likely results in a slower generation speed than Geant4 (at least on a single CPU). Therefore, as a first step to developing a geometry-independent point cloud generative model for calorimeter showers, we use the simple model architecture presented here and leave the exploration of more complex models to future work.

6.2.2 EPiC Encoder

During training, the latent space $\boldsymbol{z}$ for the conditioning of the diffusion model is created by the EPiC Encoder model. This model is a permutation-invariant VAE-like encoder, which uses three equivariant point cloud (EPiC) layers (see Chapter 7 and Reference [2]), average and summation pooling as permutation invariant pooling functions, and three fully connected layers to encode calorimeter showers into a mean and variance vector, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ respectively.

The latent space $\boldsymbol{z}$ is sampled using the reparameterization trick (see Section 4.2.1). The encoder is conditioned on the number of points N and the incident energy E. Each layer uses residual connections [254], has a hidden dimensionality of 128, and the latent space size is set to 256. The hidden dimensionality is the same as used in the EPiC-GAN [2] and the latent size is the same as suggested in Reference [251]. To achieve a close-to-normal distributed latent space, the Kullback-Leibler divergence (KLD) loss L_{KLD} is used:

$$L_{\text{KLD}} = D_{\text{KL}}(\mathcal{Z}||\mathcal{N}(0,1)) = -\frac{1}{2}\left(1 + \log(\boldsymbol{\sigma}^2) - \boldsymbol{\mu}^2 - \boldsymbol{\sigma}^2\right), \tag{6.1}$$

with the natural logarithm and the latent space given by $\boldsymbol{z} \sim \mathcal{Z} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$.

Just like in the training of a VAE and the training of the BIB-AE, the L_{KLD} is combined with the reconstruction loss (namely the diffusion model loss $L^{(t-1)}_{\text{diffusion}}$ at a random time step t) and scaled with a weighting hyperparameter β_{KLD}. As seen in Chapter 5, the choice of β_{KLD} has a large implication on the information content (the scale of L_{KLD}) encoded in the latent space $\boldsymbol{z}$. Therefore, as an advancement over the BIB-AE model loss, we introduce the minimum KLD m_{KLD} as a novel hyperparameter that clips L_{KLD} to prevent posterior collapse — i.e. the collapse of L_{KLD} to zero and therefore a latent space without any information. The training loss for the PointWise Net diffusion model and the EPiC Encoder is hence given by:

$$L^{(t-1)}_{\text{total}} = \beta_{\text{KLD}} \cdot \max(L_{\text{KLD}}, m_{\text{KLD}}) + L^{(t-1)}_{\text{diffusion}}, \tag{6.2}$$

with $\beta_{\text{KLD}} = 10^{-3}$ and $m_{\text{KLD}} = 1.0$ nats.

6.2.3 Latent Flow

For generating new data, we need to sample novel latent space encodings $\boldsymbol{z}$. Therefore, a separate model is trained simultaneously with the EPiC Encoder and the PointWise Net. This normalizing flow model, dubbed *Latent Flow*, learns to replicate the encoded latent space and is conditioned on both N and E. This flow model represents an advancement over the kernel density estimator (KDE) latent space sampling introduced in Chapter 5 since it allows for conditional generation. A theoretical introduction into normalizing flows and the specific coupling flows used for this model is given in Section 4.3.

The Latent Flow consists of ten coupling blocks with monotonic rational-quadratic splines [255]. Each spline is modeled by a two-layer neural network with LeakyReLU activation functions and a hidden dimensionality of 128. It is trained with the Adam optimizer and the negative log-likelihood (NLL) loss (see Section 4.3). Disentangling the NLL loss from the loss given in Equation 6.2, proved to be a more stable optimization regime than combining the losses as done in Reference [251].

The PointWise Net, the EPiC Encoder, and the Latent Flow are simultaneously trained[3] for 800k iterations with a batch size of 256. The learning rate is set to $2 \cdot 10^{-3}$ for the first 300k iterations and afterward reduced with a linear schedule targeting 10^{-4} at 2M iterations. All three models are implemented in PyTorch [173]. Additionally, the Latent Flow uses the nflows library [256].

[3]The presented training of CaloClouds was performed by Anatolii Korol.

6.2.4 Shower Flow

As the Latent Flow and the PointWise Net are conditioned on N, a separate model is needed during inference to produce this conditioning. This model is also conditioned on E and implemented as another normalizing flow, termed *Shower Flow*. The Shower Flow generates shower observables used for conditioning and calibration of the generated calorimeter point clouds. Overall it generates the total number of points per layer N, the relative number of points per layer $N_{z,i\in[1,30]}$, the total visible energy E_{sum}, the relative visible energy per layer $E_{z,i\in[1,30]}$, and the center of gravity in X- and Y-direction. The model architecture of the Shower Flow consists of ten normalizing flow blocks with each containing seven coupling layers — six with affine transformations [206], and one with rational-quadratic splines [255].

The Shower Flow is implemented[4] with PyTorch using the Pyro library [257]. It is trained separately from the other model parts for 350k iterations with a batch size of 2048 using the Adam optimizer with a learning rate of 10^{-5}. For the CaloClouds model presented in Reference [3], a simplified version of the Shower Flow is used, which produced only the number of points per layer and the total visible energy. The Shower Flow version discussed here is introduced in Reference [5] together with an improved calibration pipeline explained below. To be consistent across all models compared in Section 6.4, we apply this Shower Flow in CaloClouds as well.

6.2.5 Sampling & Calibration

The pipeline for sampling from the CaloClouds model is shown in Figure 6.2b and the pipeline for sampling from the CaloClouds II model in Figure 6.4b.

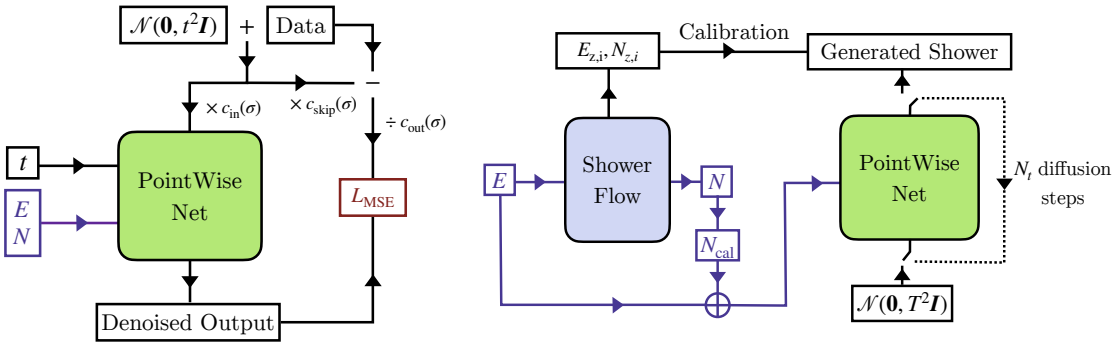
For a given incident energy E, the Shower Flow generates the relative layer-wise visible energies $E_{z,i}$, the relative layer-wise number of points $N_{z,i}$, the total number of points N, the total visible energy E_{sum}, and the center of gravities in X and Y direction, $m_{1,X}$ and $m_{1,Y}$. Note that in the original CaloClouds model introduced in Reference [3], the Shower Flow generated a reduced set of observables and the calibration pipeline deviates slightly. For consistency between both models, the Shower Flow and calibration procedure from CaloClouds II is used here for CaloClouds as well.

Using the Shower Flows' generated number of points N directly to generate this number of points in the photon shower leads to an overestimation of the number of binned cell hits N_{hits} because the points are generated a bit too spread out. Therefore, a calibration factor is applied to scale down the N points by roughly 80% to achieve an accurate N_{hits} distribution. For a precise N-dependent estimation of this calibration factor, we define it as:

$$c = p_{\text{gen}}(p_{\text{data}}(N_{\text{uncal, gen}})), \tag{6.3}$$

where p_{gen} is a cubic polynomial fit of the ratio of the number of generated cell hits $N_{\text{hits, gen}}$ (projected without calibration) to the uncalibrated number of generated points $N_{\text{uncal, gen}}$ (direct output of the Shower Flow) and p_{data} is a cubic polynomial fit of the ratio of the uncalibrated number of data points $N_{\text{uncal, data}}$ to the number of data cell hits $N_{\text{hits, data}}$. The

[4]The Shower Flow was implemented by Anatolii Korol.



(a) Training **(b)** Sampling

Figure 6.4: Overview of the training and sampling pipelines of the CaloClouds II model. During sampling with CaloClouds II (CM), just one denoising step is performed. Figure originally published in Reference [5].

calibrated number of points $N_{\text{cal, gen}}$ used for generating the calorimeter point cloud (and conditioning of the Latent Flow and PointWise Net) is then given by $N_{\text{cal, gen}} = c \cdot N_{\text{uncal, gen}}$. For the remaining text, we set $N \equiv N_{\text{cal, gen}}$.

This scaled N is used to sample the latent space $\boldsymbol{z}$ then used in turn in conjunction with N and E_{inc} as conditioning for the diffusion model PointWise Net. A total of 100 function evaluations are used to denoise a sampled noise point cloud into a calorimeter shower.

Afterward, multiple calibration steps are applied to refine the generated showers. First, all generated points are ordered by their Z-coordinate, and starting from the first layer $i = 1$, the first $N_{z,i=1}$ are set to the physical position of the center of the first layer, i.e. at $Z = 1811.5$ mm. This procedure is sequentially performed until the last layer $i = 30$ at $Z = 2010.3$ mm. Second, the total energy per layer $E_{z,i}$ is calibrated to match the one generated by the Shower Flow. Third, the center of gravities $m_{1,X}$ and $m_{1,Y}$ are calculated and calibrated to match the Shower Flow generated ones. The resulting calibrated calorimeter point clouds are compared to the simulated Geant4 showers (both after projection to the irregular grid cell structure).

6.3 CaloClouds II Model

The CaloClouds II model is an evolution of the CaloClouds model and improves it on multiple frontiers with the goal of increased generation speed without compromising generative fidelity:

1. The DDPM [214] discrete-time diffusion paradigm is replaced with the more advanced continuous-time EDM[5] [223] diffusion framework based on score matching through stochastic differential equations (SDEs) (see Section 4.4.2). This allows the usage of various ODE and SDE solvers with a variable number of function evaluations

[5]The term *EDM* is derived from the title of Reference [223] "Elucidating the Design Space of Diffusion-Based Generative Models".

without retraining, essentially trading sampling speed for generative fidelity. For CaloClouds II, this leads to fewer function evaluations during generation without a loss in generative fidelity.

2. The size of the latent space is set to $z = 0$ virtually removing the EPiC Encoder and the Latent Flow from the training and sampling pipeline, as we did not observe a noticeable difference with a latent space $z > 0$ with CaloClouds II. This improves the sampling speed.

3. The already mentioned improvements to the calibration procedure involving a new Shower Flow are implemented.

4. We apply consistency distillation to distill the EDM diffusion model in CaloClouds II into a consistency model [219] (see Section 4.5). The resulting model, dubbed CaloClouds II (CM), can be used for single-shot sampling, greatly improving the sampling speed on both CPU and GPU.

An overview of the training and sampling with the CaloClouds II model is given in Figure 6.4. The conditioning of the PointWise Net is the same as in CaloClouds. During sampling, the same calibration procedure as in the previous Section 6.2 is performed. The CaloClouds II EDM diffusion model is implemented using PyTorch [173] with the same PointWise Net architecture. For the parameterization of the score-based EDM diffusion model training, we follow the suggestions given in Reference [223]. With a batch size of 128, the model was trained for 2M iterations using the Adam optimizer and a fixed learning rate of 10^{-4}. As the final model, an exponential moving average (EMA) of model weights over the whole training is used. For sampling, we follow Reference [223] and use the Heun 2^{nd}-order Runge-Kutta ODE solver without any noise injection, essentially sampling it as a continuous normalizing flow (CNF) (see Section 4.3.2). For the step size parameterization, we follow Reference [223] as well. With $T = 80$ ($\sigma_{\max}$) and $\epsilon = 0.002$ ($\sigma_{\min}$), we found that below 25 function evaluations (13 Heun ODE solver steps, as the final step is only a 1^{st}-order Euler step) the performance degrades.

To further speed up the shower generation, we distill the EDM diffusion model into a consistency model (CM). Following Reference [219], we use $n = 18$ Heun ODE solver steps as the baseline to perform the consistency distillation. The distillation training is done for 1M iterations at a batch size of 256 with the Adam optimizer. Note that compared to progressive distillation [218], only a single training is necessary to achieve a model capable of single-shot sampling. Furthermore, the resulting CaloClouds II (CM) model can be used for single-shot and multi-step generation. We found the single-step generation on par with the fidelity of CaloClouds II leading to a significant speed-up of the shower generation.

6.4 Photon Shower Generation

We benchmark the generative fidelity of the CaloClouds model variants — CaloClouds, CaloClouds II, and CaloClouds II (CM)— by generating many photon showers with uniformly distributed energy conditioning as well as fixed energy showers. These photon

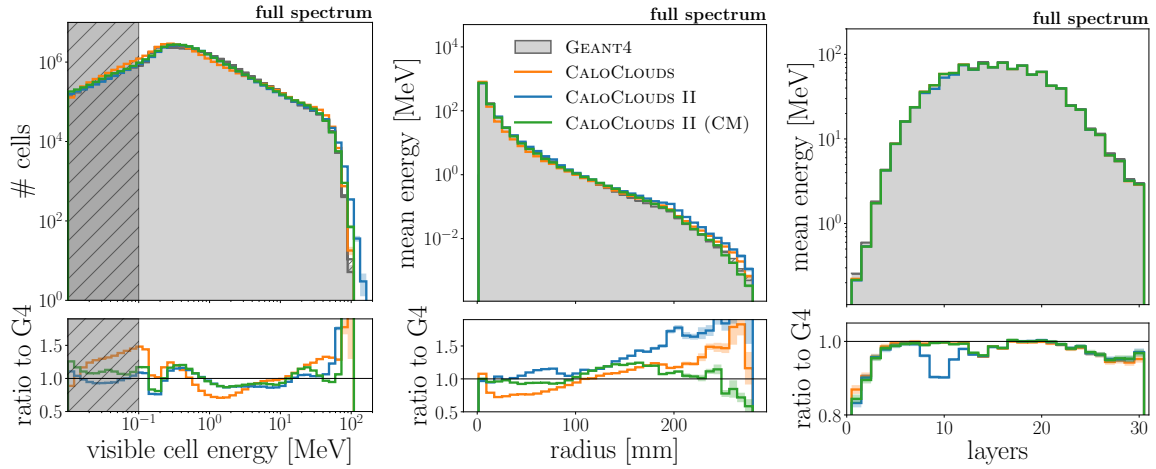


Figure 6.5: Histogram of the cell energies (left), radial shower profile (center), and longitudinal shower profile (right) for GEANT4, CALOCLOUDS, CALOCLOUDS II, and CALOCLOUDS II (CM). In the cell energy distribution, the region below the half-MIP threshold of 0.1 MeV is grayed. In all three plots, 40,000 showers are shown, sampled from a uniform distribution of incident particle energies between 10 and 90 GeV. The bottom panels provide the ratio between the generative models and GEANT4. The error band corresponds to the statistical uncertainty in each bin. Figures originally published in Reference [5].

shower point clouds are projected back into the real ILD ECAL cell geometry, including realistic cell gaps and staggering. In Section 6.4.1 we judge the performance with histograms of various physics observables, while we use Wasserstein distance-based evaluation scores in Section 6.4.2. Additional classifier scores are shown in Section 6.4.3 using a high-level fully-connected classifier. In Section 6.4.4 we compare the generation speed of all three models both on CPU and GPU to the GEANT4 baseline (CPU only).

6.4.1 Physics Performance

To evaluate the generative fidelity of all three CALOCLOUDS models on a uniform energy spectrum (10 – 90 GeV), we generate with each model 40,000 showers and compare histogram of various shower observables to the GEANT4 baseline. In Figure 6.5 we show the distribution of cell energies (left), the radial shower profile (center), and the longitudinal shower profile (right). In the cell energy spectrum, a threshold at 0.1 MeV is shown. This is the applied half-MIP cut which is used for all high-level observables, since below this threshold the cell energy response is indistinguishable from electronic noise. All models represent the spectrum well, but the CALOCLOUDS II models model the bulk of the distribution a bit better than CALOCLOUDS, yet overestimate the maximum cell energy slightly.

The radial shower profile, i.e. the mean distribution of energy in a concentric region of a certain radius around the shower incident axis, is also modeled well by all three models, yet the best in particular at high and very large radii is the CALOCLOUDS II (CM) model.

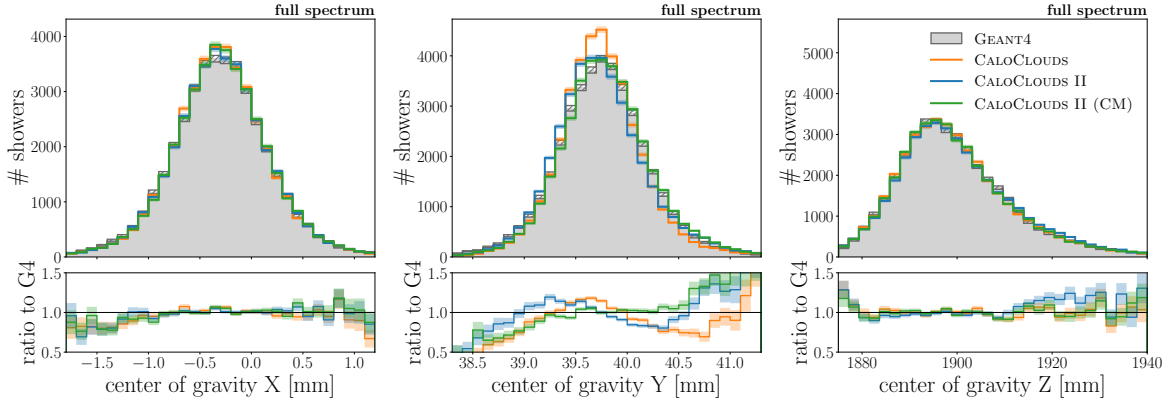


Figure 6.6: Center of gravity distributions along the X- (left), $Y-$ (center), and $Z-$axis (right) of the GEANT4 and model generated photon showers. 40,000 showers are shown, sampled from a uniform distribution of incident particle energies between 10 and 90 GeV. The error band corresponds to the statistical uncertainty in each bin. Figures originally published in Reference [5].

This radial profile is not affected by the layer-wise energy and number of hits calibration and is therefore a particularly good benchmark for the diffusion model itself without the post-diffusion calibration pipeline.

All three models underestimate the mean energy in the first few layers, but model well the remaining ones, except CALOCLOUDS II around layer 10. This longitudinal shower profile is largely determined by the Shower Flow, as it generates the layer-wise energies. A sequence of higher and lower energy depositions is visible in the center of the distribution. This is because (for technical reasons) every second tungsten absorber layer is sandwiched between two active silicon layers each facing this respective absorber. The result is an observable pair-wise difference in the sampling fraction of consecutive layers.

Figure 6.6 shows the center of gravity (the energy-weighted center of the shower or the first moment) of the photon showers along the X- (left), Y- (right), and Z-axis (center) — for a given axis, the energy is summed up along the other two. In the X- and Y-direction, these distributions are affected by the center of gravity calibration on point-level, although slight deviation can occur due to the projection to the cell level. All three model very well the X- and Y-direction, while for the Y-direction, the spectra deviate a bit with the CALOCLOUDS model producing a too narrow distribution. The challenge in the Y-axis is due to the cell staggering, which is layer-dependent in the Y-direction, but uniform in the X-direction. Additionally, the simulated magnetic field affects the showers in Y-direction.

Next, we explore single incident samples. For set energies at 10, 50, and 90 GeV, 2,000 showers were generated with each model. Figure 6.7 shows their total visible energy (left) and their number of hits distribution (right). All three generative models reproduce the energy sum well. The number of hits distribution is particularly challenging as during the projection into cell-level multiple points are binned together. Overall the modeling is accurate, yet small mismodelings are still visible in particular for 90 GeV, where CALOCLOUDS II is best, CALOCLOUDS too broad, and CALOCLOUDS II (CM) overestimates the number of hits.

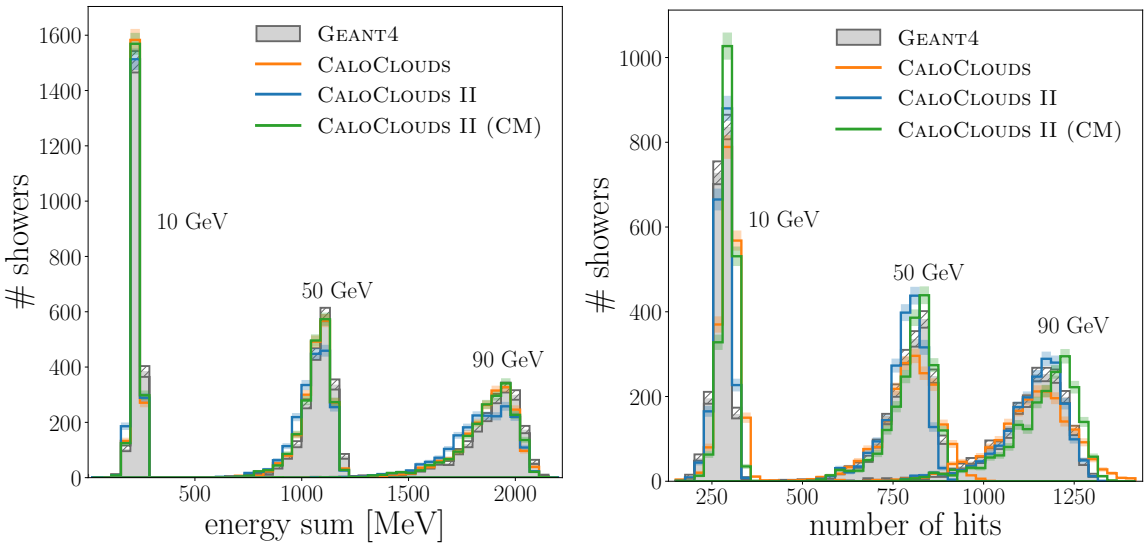


Figure 6.7: Total visible energy (left) and the number of cell hits (right) for fixed energy showers at 10, 50, and 90 GeV. 2,000 showers are shown for each model and energy. The error band corresponds to the statistical uncertainty in each bin. Figures originally published in [5].

These mismodelings at large energies are due to the polynomial fit used in the number of points scaling. The fit does not perform well on the edges of the energy range. Were the energy range expanded, we would expect a higher fidelity for the 10 and 90 GeV photon showers.

In summary, the physics performance of all three CaloClouds variants is good. In particular, for the radial distribution, it is visible that the CaloClouds II models are slightly better than CaloClouds, yet their main advantage lies in their improved computational efficiency.

6.4.2 Evaluation Scores

To gain a more precise understanding of the generative fidelity of the three models compared to Geant4, we calculate multiple evaluation scores based on the distributions shown in Section 6.4.1. In line with publications such as References [2,62,240], we use the 1-Wasserstein distance W_1 to compare 1-dimensional distributions. The advantage of the Wasserstein distance is, that it is trivial to calculate in one dimension and that it is an unbinned estimator, so no parameter choices have to be made in the distance calculation. A disadvantage is, that it does not always align well with results gained from density-based measures, such as the Kullback-Leibler divergence or the visual inspection of histograms, and it is prone to outliers. Therefore, these scores can give a good indication of a well-performing model, but should always be considered in conjunction with other evaluation methods such as histograms.

For these distributions, we consider the number of cell hits N_{hits}, the sampling fraction $E_{\text{vis}}/E_{\text{inc}}$, the cell energy E_{cell}, the center of gravity in X-, Y-, and Z-direction $m_{1,\alpha\in\{X,Y,Z\}}$

Table 6.2: Evaluation of the three CaloClouds variants in comparison to Geant4 using 1-Wasserstein distance-based scores for various standardized shower and cell level observables. The values are the mean and standard deviation of 10 calculated scores comparing 50k Geant4 and 50k generated showers each. Table originally published in Reference [5].

Simulator	$W_1^{N_{\text{hits}}}$ ($\times 10^{-3}$)	$W_1^{E_{\text{vis}}/E_{\text{inc}}}$ ($\times 10^{-3}$)	$W_1^{E_{\text{cell}}}$ ($\times 10^{-3}$)	$W_1^{E_{\text{long}}}$ ($\times 10^{-3}$)	$W_1^{E_{\text{radial}}}$ ($\times 10^{-3}$)	$W_1^{m_{1,X}}$ ($\times 10^{-3}$)	$W_1^{m_{1,Y}}$ ($\times 10^{-3}$)	$W_1^{m_{1,Z}}$ ($\times 10^{-3}$)
Geant4	0.7 ± 0.2	0.8 ± 0.2	0.9 ± 0.4	0.7 ± 0.8	0.7 ± 0.1	0.9 ± 0.1	1.1 ± 0.3	0.9 ± 0.3
CaloClouds	$\mathbf{2.5 \pm 0.3}$	11.4 ± 0.4	15.9 ± 0.7	$\mathbf{2.0 \pm 1.3}$	38.8 ± 1.4	4.0 ± 0.4	8.7 ± 0.3	1.4 ± 0.5
CaloClouds II	3.6 ± 0.5	26.4 ± 0.4	$\mathbf{15.3 \pm 0.6}$	3.7 ± 1.6	11.6 ± 1.5	$\mathbf{2.4 \pm 0.4}$	$\mathbf{7.6 \pm 0.2}$	3.9 ± 0.4
CaloClouds II (CM)	6.1 ± 0.7	$\mathbf{9.8 \pm 0.5}$	16.0 ± 0.7	$\mathbf{2.0 \pm 1.4}$	$\mathbf{8.3 \pm 1.9}$	3.0 ± 0.4	9.5 ± 0.6	$\mathbf{1.2 \pm 0.5}$

as well as ten longitudinal and radial energy observables, $E_{\text{long},i\in[1,10]}$ and $E_{\text{radial},i\in[1,10]}$, based on the longitudinal and radial shower profiles in Figure 6.5. These ten longitudinal (radial) observables are calculated with the energy depositions clustered together in ten equiprobable bins of consecutive layers (concentric regions), i.e. the bin edges (layer-wise or radial) of each observable is determined such that each observable is calculated with the same statistics. Histograms of the energy observables and their bin edges can be found in Appendix A.

A total of 500,000 photon showers are generated with the generative models as well as with Geant4 with a uniformly distributed incident energy between 10 and 90 GeV. Each Wasserstein score is ten times calculated using batches of 50,000 showers. The Geant4 scores are calculated the same way, using not-overlapping batches of Geant4 vs. Geant4. For the $W_1^{E_{\text{cell}}}$ score, only the first 50,000 cell hits are used, since using all hits from all 50,000 showers would be computationally impractical. For the radial and longitudinal observables, we quote the mean value of the respective set of observables, E_{radial} and E_{long}. To achieve overall comparable scores, each observable is standardized, i.e. their mean and standard deviation are shifted to zero and one. Tabel 6.2 shows the resulting scores with mean and standard deviation over the ten batches.

The scores paint a very similar picture to the histograms shown in Section 6.4.1, namely all models perform similarly, yet all deviate from the Geant4 truth. The largest difference between the models is observable for the $W_1^{E_{\text{radial}}}$ score, there the CaloClouds II (CM) model performs best and the CaloClouds model quite a bit worse than both CaloClouds II and CaloClouds II (CM). This is in line with what can be seen in Figure 6.5 (center) as well as in the individual radial observables shown in Figure A.1 of Appendix A. Overall, all three models produce high-fidelity photon showers, tho further research is needed to exactly match the Geant4 baseline.

6.4.3 Classifier Scores

Since the Wasserstein scores only compare 1-dimensional distributions and are complex to compute for multiple dimensions, we use as a third evaluation method the *classifier score*. This classifier score is given by the area under the receiver operating characteristic curve (AUC) score of a binary classifier trained to separate "real" Geant4 showers from "fake"

Table 6.3: Comparison of the CaloClouds model performance to Geant4 with the area under the receiver operating characteristic curve (AUC) score. The AUC score was calculated using a high-level binary classifier with shower observables as input trained to separate Geant4 from CaloClouds generated showers. The values presented are the mean and standard deviation of ten AUC scores with each of the ten classifiers trained and evaluated with a different (not overlapping) dataset split. Table originally published in Reference [5].

Simulator	AUC
CaloClouds	0.999 ± 0.001
CaloClouds II	0.928 ± 0.001
CaloClouds II (CM)	$\mathbf{0.923 \pm 0.001}$

generated showers. For HEP applications, the classifier score and its variants were explored in detail in Reference [241]. In general, the classifier score (i.e. fooling the classifier with AUC = 0.5) could be seen as the "gold standard", that one should strive to achieve with a generative model. However, it is not trivial to develop a generative model that achieves anything but AUC = 1.0. AUC values between 0.5 and 1.0 are difficult to interpret, yet can indicate which model might perform better.

To calculate the score, we use a high-level classifier with a fully connected network separating real and fake showers based on the shower observables introduced in Section 6.4.2, namely the number of hits, sampling fraction, the three center of gravity observables, ten radial observables, and ten longitudinal observables. An alternative would have been to use a low-level classifier directly on a cell-level point cloud, but such a classifier would need to be permutation-equivariant and work with $\mathcal{O}(1000)$ points and is therefore difficult to implement. Additionally, we have seen in Section 5.5 that a high-level classifier offers better separability between multiple models than a low-level classifier.

As a classifier[6], we use a three-layer fully connected network (32, 16, 8 nodes) with LeakyReLU activations and an output Sigmoid activation. The classifier is trained for 10 epochs using the Adam optimizer and the binary cross-entropy loss. Out of the ten epochs, the epoch with the lowest validation loss is chosen for the AUC evaluation on the test set. For the training, validation, and test dataset split, we apply 80%, 10%, and 10% on the full set containing 500,000 Geant4 and 500,000 generated showers. To estimate an error in the classifier score, the classifier was trained ten times for each generative model, each using a different data split resulting in ten different (not overlapping) test and validation sets.

For all three models, we show in Table 6.3 the mean and standard deviation of the AUC scores over the ten trainings. Based on the high-level observables, the classifier can perfectly distinguish between CaloClouds generated and Geant4 simulated showers (AUC $\approx$ 1.0). The showers generated with CaloClouds II already achieve a better score with AUC = 0.928, and the CaloClouds II (CM) model is best with a score of AUC = 0.923. This result can be explainable by the better generation of the radial energy observables by the

[6]The classifier was implemented by Anatolii Korol.

Table 6.4: Computational efficiency of CaloClouds, CaloClouds II, and CaloClouds II (CM) in comparison to the baseline Geant4 simulator on a single core of an Intel® Xeon® CPU E5-2640 v4 (CPU) and on an NVIDIA® A100 with 40 GB of memory (GPU). Per run 2,000 showers were generated with incident energy uniformly distributed between 10 and 90 GeV. The values are the means and standard deviations over 10 runs. The number of function evaluations (NFE) indicates the number of diffusion model passes. Table originally published in Reference [5].

Hardware	Simulator	NFE	Batch Size	Time / Shower [ms]	Speed-up
CPU	Geant4			3914.80 ± 74.09	×1
	CaloClouds	100	1	3146.71 ± 31.66	×1.2
	CaloClouds II	25	1	651.68 ± 4.21	×6.0
	CaloClouds II (CM)	1	1	84.35 ± 0.22	×46
GPU	CaloClouds	100	64	24.91 ± 0.72	×157
	CaloClouds II	25	64	6.12 ± 0.13	×640
	CaloClouds II (CM)	1	64	2.09 ± 0.13	×1873

CaloClouds II variants, as we already observed in the previous sections. Hence, we find that CaloClouds II constitutes an improvement over the previous CaloClouds model, but more research is needed to achieve a model able to produce samples on a level of fidelity able to fool the high-level classifier.

6.4.4 Timing Comparison

We observe high fidelity by all three CaloClouds models, yet they do not achieve an accuracy exactly on par with Geant4. However, depending on the application, i.e. proof of principle studies or large parameter scans, a slight loss in performance might be acceptable if it comes with the advantage of greatly increased generation speed. Therefore, a detailed benchmark of the computational efficiency of each model is of interest. Since most generative models are based on deep learning libraries such as PyTorch, which are optimized for parallelizable computations on GPUs, the models allow for a straightforward generation speed-up on GPUs when compared to the Geant4 simulation, which is only available to run on CPUs. However, GPUs are generally more expensive than CPUs and not as widely available, since most of the computational infrastructure in HEP requires large numbers of CPUs. Hence, a more fair comparison between Geant4 and the CaloClouds models can be drawn when sampling on a single CPU.

In Tabel 6.4 we provide the average generation time for generating a single shower[7]. We quote the mean time of 10 runs, each generating 2,000 showers with a uniform energy distribution between 10 and 90 GeV. Note that just like Geant4 and unlike fixed grid- / image-based generative models, the generation speed of the point cloud-based CaloClouds

[7]The Geant4 benchmark was performed by Anatolii Korol.

models scales with the number of points generated and therefore also with the incident energy. We additionally provide in the table the number of function evaluations in each model, as this constitutes the source of the large speed improvements of the CaloClouds II models over the CaloClouds model. The batch sizes on GPU are optimized for computational performance and are set to 64 for all three models.

On CPU, the CaloClouds model achieves a speed-up of 1.2× over Geant4, CaloClouds II achieves a speed-up of 6.0×, and CaloClouds II (CM) achieves a speed-up of 46×, which is even about 5× faster than the BIB-AE model introduced in Chapter 5. For reference, this large of a speed-up means, that using the CaloClouds II (CM) model one can generate within a day as many showers as with Geant4 in 1.5 months. On GPU these speed-ups are naturally even larger and a speed-up of up to 1873× can be achieved using CaloClouds II (CM).

The training of the CaloClouds model on a NVIDIA® A100 GPU took about 80 hours and the training of the CaloClouds II model took around 50 hours on the same GPU. The consecutive distillation of CaloClouds II (CM) took about 100 hours. This distillation is fast compared to other distillation methods such as progressive distillation, for which multiple models have to be trained, each with half as many denoising steps as the previous.

Overall, the CaloClouds model yields already a large speed-up for generating photon showers using a GPU, while CaloClouds II and particularly CaloClouds II (CM), achieve comparable fidelity, yet with greatly improved sampling speed even on CPU.

6.5 Summary

For the simulation of particle showers in highly-granular calorimeters, generative models are being explored to accelerate this process within the currently expected computing budget. Previous fixed-size or image-based generative models, such as the BIB-AE discussed in Chapter 5 are on the one hand faster than Geant4, but on the other hand computationally inefficient as particle showers in highly-granular calorimeters are very sparse when represented as a 3-dimensional image using much of the computation to essentially produce non-existing sensor responses. Representing showers as point clouds is a way around this limitation as only existing cell hits are simulated. Additionally, the point cloud can be made up of (virtual) ultra-high granular energy depositions, effectively leading to a largely geometry-independent generative model. However, this requires the generation of high cardinality point clouds with many more points than previously explored point cloud generative models in HEP can handle.

With CaloClouds we have developed a model able to generate photon calorimeter showers as high cardinality point clouds with an unprecedented fidelity. Its core is a permutation-equivariant DDPM-based diffusion model, dubbed PointWise Net, for generating point clouds. PointWise Net is accompanied by multiple other sub-models, including the Shower Flow for conditioning and calibration, the EPiC Encoder (during training), and the Latent Flow (during generation). Together, the full CaloClouds model is able to generate high-fidelity photon showers in the IDL ECAL 20% faster than Geant4 on a single CPU and 157× faster on a GPU.

To increase the generation speed even further, we introduce CaloClouds II and its variant CaloClouds II (CM) as an evolution of the CaloClouds model, which yields better generative fidelity and offers a significant speed-up over both CaloClouds and Geant4 even on a CPU. The main speed-up is gained by moving from the DDPM-based discrete-time diffusion paradigm to the EDM-based continuous-time diffusion. This allows for the use of a more advanced sampler, reducing the number of needed function evaluations by a factor of four, and makes it possible to distill the PointWise Net into a consistency model for single shot sampling without loss in fidelity. This effectively reduces the number of diffusion model passes from 100 in CaloClouds to a single pass in CaloClouds II (CM), therefore leading to a significant speed-up. Additionally, the Latent Flow is omitted, and further calibrations are introduced in CaloClouds II.

Together, these improvements allow for the CaloClouds II (CM) model to gain an impressive speed-up of 46× over the Geant4 simulation on CPU, and for a speed-up of up to 1873× on a GPU. On CPU this is even 5× faster than the BIB-AE. Further, consistency distillation is much more computationally efficient than other distillation methods such as progressive distillation, which was previously explored for HEP generative models [60, 258], as it just needs a single training to allow for single shot generation. The CaloClouds II (CM) model constitutes the first application of a consistency model for calorimeter shower simulation.

Comparing the three CaloClouds variants based on shower observables, using both histograms and evaluation metrics, including a high-level classifier, we find slightly improved performance with CaloClouds II over CaloClouds and the best performance with CaloClouds II (CM) while being computationally the cheapest. The student model CaloClouds II (CM) outperforming the teacher model CaloClouds II appears counter-intuitive. However, it is known that errors introduced in earlier ODE solver steps of a diffusion model can propagate and negatively impact later solver steps [223]. The consistency model avoids such mismodelings with single-shot sampling. Yet, clear deviations from the Geant4 baseline are still observable. Further comparisons with other established image-based generative models such as the BiB-AE or the CaloScore models [59,60] should be conducted. To this end, the CaloClouds II model is included in the *CaloChalle nge* [14] community benchmark.

To achieve a high-fidelity generation of hadronic showers, such as charged pion showers, it may be necessary to go beyond the i.i.d. assumption and consider inter-point correlations. This could be done by using transformer (cross)-attention layers [180], fast attention layers [259], or equivariant point cloud (EPiC) layers [2] (see Chapter 7). All these methods are more computationally costly than the PointWise Net, hence the exact layer implementations, the diffusion modeling framework, and the number of denoising steps need to be carefully considered.

Currently, the CaloClouds II and CaloClouds II (CM) models are being investigated for the generation of electromagnetic showers in the CMS HGCAL, where the geometry is more complex with hexagonal tiles. The results look promising and advance the CaloClouds models by including the hit time, i.e. modeling a 5-dimensional point cloud [260].

Chapter 7

Point Cloud Modeling of Particle Jets

The results in this chapter have previously been published in References [2,4] in collaboration with Cedric Ewen, Darius A. Faroughy, Tobias Golling, Gregor Kasieczka, Matthew Leigh, Guillaume Quétant, John Andrew Raine, Debajyoti Sengupta, David Shih, and Jesse Thaler. My contribution to Reference [2] was the development and implementation of the EPiC-GAN model including the EPiC layers, the training and evaluation of the model, the majority of the manuscript writing process, the monitoring of the reviewing process, and addressing the reviewer remarks. I was part of the inception of the project leading to Reference [4], as it combines the EPiC layers of the EPiC-GAN model with diffusion models / continuous normalizing flows. The EPiC-FM model was implemented, trained, and evaluated by Cedric Ewen under my supervision as part of his master thesis [18]. Further, I wrote sections of the manuscript and addressed reviewer comments. The figures and tables in the chapter are similar or identical to the ones in the original publications.

I further contributed to applying the EPiC-FM model to anomaly detection in Reference [12] and to its application to the JetClass [11] dataset in Reference [261]. As my involvement was larger for References [2,4] and the focus in this thesis lies on the development of the generative models themself, I am focusing this chapter on these two publications.

In the previous Chapter 6, the CaloClouds models are introduced as generative models for calorimeter shower point clouds with up to 6,000 points per event. Although reaching high fidelity on relatively simple electron showers and excelling at computational efficiency, the models are not as accurate as other voxel-based generative models such as the BIB-AE discussed in Chapter 5. This limitation is likely due to the simplified assumption of sampling each point i.i.d. (independent and identically distributed) using the PointWise Net. Yet this i.i.d. assumption also allows for the computational efficiency of the models. More complex generative models based on graph-networks [62] and transformers [240, 249, 258, 262, 263] are being developed for HEP applications, yet these layer types come with a significant computational cost as they usually scale quadratically with the size of the point cloud.

In this chapter, a novel kind of neural network layer structure for point cloud data is introduced. It is based on the Deep Sets [176] principle (known in the particle physics

community as energy and particle flow networks [181]) and allows building generative models that are similarly performant as graph-network and transformer models, but with a computational cost that only scales linearly with the point cloud size. As a layer optimal for point cloud data, it models an equivariant function and works with variable-sized point clouds. We, therefore, introduce it as the *equivariant point cloud* (EPiC) layer.

We further introduce three generative models that utilize this layer: a generative adversarial network, dubbed *EPiC-GAN*, a diffusion model, dubbed *EPiC-JeDi*, and a continuous normalizing flow trained with optimal-transport flow matching, dubbed *EPiC-FM*. The models are applied to the benchmark datasets JetNet30 [264] and JetNet150 [265]. Extensive comparisons between the three models of their generative fidelity on the JetNet datasets are shown, as well as comparisons with the graph-based *message-passing GAN* (MP-GAN) [62] and the transformer-based *PC-JeDi (Point Cloud Jets with Diffusion)* [262] — both state-of-the-art jet generative models. We find that the EPiC generative models allow for high-fidelity jet generation while being significantly more resource-efficient than the graph- and transformer-based models. Further, the EPiC layers contain a global latent space allowing for a vector to interpret the physical quantities learned by the models.

The EPiC-GAN is particularly resource-efficient but is surpassed in generative fidelity by the EPiC-JeDi and EPiC-FM models, which can both be viewed as continuous normalizing flows (CNFs) [207] during sampling. As they need $\mathcal{O}(100)$ model passes for sampling a single jet, this comes however at an increased computational cost. Both EPiC-JeDi and EPiC-FM use the same neural network architecture but with two distinct training objectives: DDIM[1]-based score matching [266] (EPiC-JeDi) or optimal-transport flow matching [212] (EPiC-FM). This allows us to directly compare these two objectives for training CNFs.

The JetNet dataset consists of particle constituents of jets, reconstructed from a simplified LHC detector simulation. When generating jets like these with a generative model, one would bypass the complete simulation chain, including the generation of the hard process, the parton showering, the hadronization, the detector simulation, and the reconstruction of the jets. This would greatly speed up the simulation of particle collisions, however, one would need to train a generative model for every single SM and BSM process.

Much simpler is the replacement of individual steps in the simulation chain, such as the event generation or the detector simulation, with a generative model. Of these, the detector simulation, i.e. the calorimeter response, is the most time-consuming step and relatively easy to replace, as mostly electrons and pions interact with the calorimeter. Hence, an accurately trained generative model only trained on electrons and pions would be sufficient to replace a large part of the detector simulation. This introduces another problem, that the simulated objects in calorimeters are much larger than the number of reconstructed jet constituents $\mathcal{O}(1,000 - 10,000)$ vs. $\mathcal{O}(100)$. Such a model is proposed with the CaloClouds model family in the previous Chapter 6.

For training generative models that replace either parts of or the whole simulation chain, an ideal dataset would consist of separate but consistent sub-datasets stored after each part of the chain. This way one could investigate which part is easier to replace, or whether it is possible to replace the whole chain.

[1]The term *DDIM* is derived from the title of Reference [266] "Denoising Diffusion Implicit Models".

As it stands the JetNet dataset is an interesting R&D laboratory for the development of generative models in particle physics, but not a realistic application for such a model. It is well suited for the evaluation and optimization of different point cloud generative models, e.g. how they model particle interactions and their training objectives. Since its release, several generative models were evaluated on JetNet, including GANs [2, 62, 240, 267, 268], normalizing flows [4, 269], and diffusion models [258, 262, 263].

The JetNet datasets used for the studies in this chapter are introduced in Section 7.1. The EPiC-GAN model is introduced in Section 7.2 and its evaluation on both fidelity and computational efficiency is presented in Section 7.3. We further introduce the EPiC-FM and EPiC-JeDi models in Section 7.4 and compare them to both the EPiC-GAN and PC-JeDi in Section 7.5. A summary concludes this chapter in Section 7.6.

7.1 Jets & Particle Clouds

Jets consist of a collection of particles, the jet constituents, with each particle i being defined for example as a 4-momentum vector $\boldsymbol{p}_i = (E_i, p_{x,i}, p_{y,i}, p_{z,i})$ or in collider coordinates $(p_{\mathrm{T},i}, \eta_i, \phi_i)$ (transverse momentum, pseudorapidity, and azimuthal angle) — when assuming massless particles. This set of particles can be represented as a point cloud $P = \{\boldsymbol{p}_i\}_{i=1}^{N}$, where N is the number of particles in the jet (the particle multiplicity).

To benchmark the performance of the models introduced in this chapter, we use the JetNet30 [264] and JetNet150 [265] datasets which are specifically designed to study the performance of jet generative models and were introduced in Reference [62]. The jets in the datasets were first simulated for studying the impact of calorimeter effects on jet substructure observables of highly-boosted jets in Reference [270] and published as the "HLS4ML LHC Jet dataset (30 particles)" [271].

As outlined in Reference [270], the events are produced from simulated proton-proton collisions at a center-of-mass energy of $\sqrt{s} = 13$ TeV. Highly-boosted parton-level top ($t\bar{t}$), light quark ($q\bar{q}$), and gluon (gg) events are generated at leading-order using MadGraph5_aMC@NLO [272] (version 2.3.1). The parton and gauge boson energies are centered around 1 TeV with a window of $\delta p_{\mathrm{T}}/p_{\mathrm{T}} = 0.01$. Their decay and showering are simulated with PYTHIA [115] (version 8.212) using the Monash 2013 tune [273]. After hadronization, the samples were passed through a custom parametric simulation of a LHC detector which considers real detector effects by smearing and granularization of the particle level features [270, 274].

All jets are clustered using the anti-k_{T} algorithm [275] with a radius parameter of $R = 0.4$[2] using FastJet [276] version 3.1.3. Afterward a jet p_{T} cut of 0.8 TeV $< p_{\mathrm{T}} <$ 1.6 TeV is applied to remove outlier events since the jet p_{T} spectrum is broadened by kinematic recoil and energy migration in and out of the jet cone.

This generation procedure resulted in about 170,000 particles per jet class. To create the JetNet30 and JetNet150 datasets, additional processing is applied to the samples as outlined in

[2] In Reference [270] the clustering radius was incorrect stated as $R = 0.8$, which was subsequently propagated to various later publications, including Reference [62]. From observing the η and ϕ distributions as well as the W-peak in the jet mass distribution, we deduce that the actual radius must be $R = 0.4$.

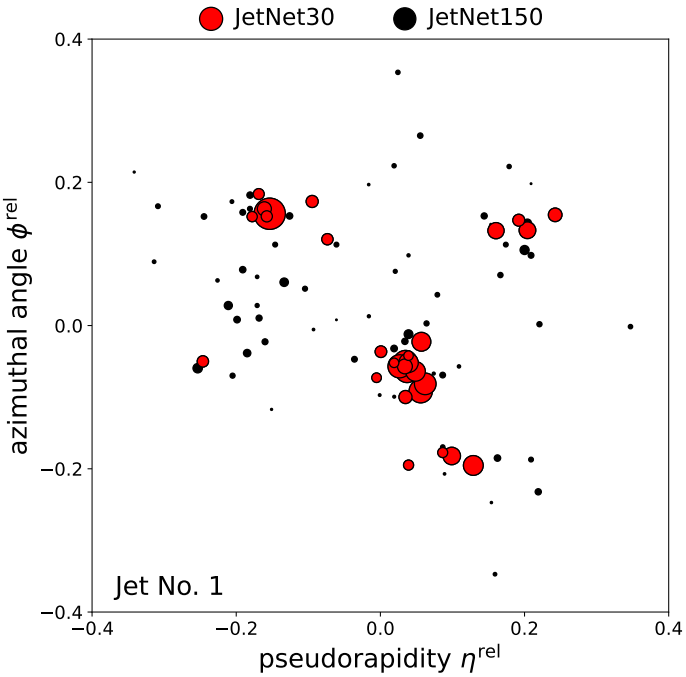


Figure 7.1: Visualization of a single top jet event in the JetNet dataset. Each point represents a jet constituent. The size of the point scales with the particle p_{T}. The first 30 jet constituents are used in the JetNet30 dataset (red). All 96 jet constituents are used in the JetNet150 dataset (red & black).

Reference [62]: The jet constituents (particles) are normalized, centered, and represented with three particle features, the relative transverse momentum[3] $p_{\text{T}}^{\text{rel}} = p_{\text{T}}^{\text{particle}}/p_{\text{T}}^{\text{jet}}$, the relative pseudorapidity $\eta^{\text{rel}} = \eta^{\text{particle}} - \eta^{\text{jet}}$, and the relative azimuthal angle $\phi^{\text{rel}} = \phi^{\text{particle}} - \phi^{\text{jet}}$. Afterward, the particles are p_{T} ordered and a particle multiplicity cut is applied. A cut at 30 particles is performed for the JetNet30 datasets and at 150 particles for the JetNet150 datasets. For model training and evaluation we use a 70%/15%/15% dataset split — about 120,000 training events and about 25,000 validation and test events.[4] A visualization of the same single top jet in JetNet30 and JetNet150 is shown in Figure 7.1.

7.2 EPiC-GAN

When EPiC-GAN was introduced, the state-of-the-art model on the JetNet30 dataset was a graph neural network, the message-passing GAN (MP-GAN) [62], using layers based on fully-connected message-passing neural network (MPNN) blocks [178] (illustrated in Figure 3.4b). While the MPGAN achieves very good generative fidelity on all three JetNet30 datasets (gluons, light quarks, and tops), it is rather slow to evaluate considering that fully-connected

[3] In Reference [62], the assumption $p_{\text{T}}^{\text{jet}} \approx \sum_i p_{\text{T},i}^{\text{particle}}$ was made and the terminology in this chapter follows that assumption.

[4] Note that the validation set is only used for the model choice of the EPiC-GAN. For the evaluation of the continuous time generative models, we combine the validation and test sets.

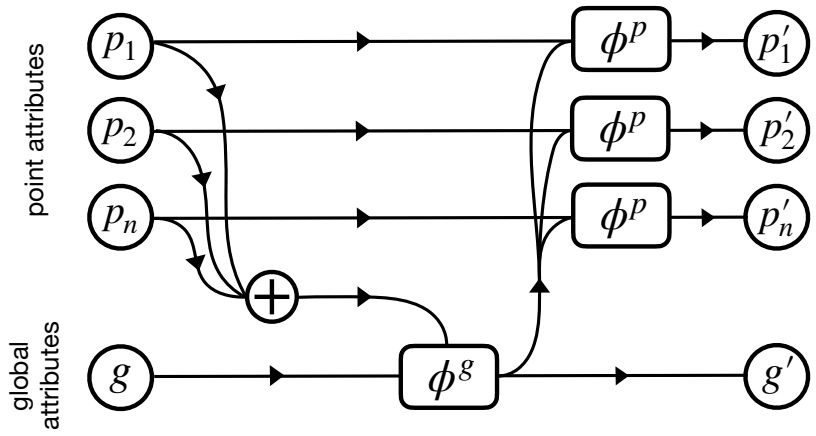


Figure 7.2: Structure of the equivariant point cloud (EPiC) layer. The global function ϕ^g and point function ϕ^p are implemented as two-layer neural networks. The $\oplus$ symbol describes the (permutation-invariant) aggregation function $\rho^{p\to g}$ with both element-wise summation and average pooling. Figure originally published in Reference [2].

MPNN blocks scale quadratically with the number of input points. Therefore, no results were published for this model (or by any other model previous to the EPiC-GAN) on JetNet150.

In this section, we are introducing the *equivariant point cloud* (EPiC) layers, which we developed to provide a minimal layer structure that is still able to account for inter-point correlations. We then outline the generator and discriminator architecture of the *equivariant point cloud generative adversarial network* (EPiC-GAN), which largely consists of EPiC layers. Further information on the EPiC-GAN training and sampling is detailed below. The code for the EPiC layers and the EPiC-GAN as well as the trained weights are publicly available[5].

7.2.1 Equivariant Point Cloud (EPiC) Layer

In contrast to MPNN layers, the computations performed in the EPiC layers scale linearly with the number of points as they are based on the DeepSets framework [176]. Adapting the notation in Reference [177], a 2-tuple point cloud is defined as $C = (\boldsymbol{g}, P)$ (a graph without edge features). As defined above, P is a set of points $\boldsymbol{p}_i$ and $\boldsymbol{g}$ represents the global attributes of the whole point cloud, e.g. the p_T of a jet or particle multiplicity N.

The structure of the EPiC layers is outlined in Figure 7.2. Transformation of both the global attributes $\boldsymbol{g} \to \boldsymbol{g}'$ and the points $P \to P'$ are performed with two consecutive computations:

$$\boldsymbol{g}' = \phi^g(\boldsymbol{g}, \rho^{p\to g}(P)), \tag{7.1}$$

$$\boldsymbol{p}'_i = \phi^p(\boldsymbol{g}', \boldsymbol{p}_i). \tag{7.2}$$

Both the global function ϕ^g and point function ϕ^p are learned by a 2-layer fully connected network with LeakyReLU activation functions. To achieve permutation-equivariance with respect to the points P, the aggregation function $\rho^{p\to g}$ is a concatenation of both element-wise

[5]Code and weights available on GitHub: `https://github.com/uhh-pd-ml/EPiC-GAN`

summation and average pooling — both permutation-invariant operations. It maps the point features into a common global feature space. For this global space to contain information about the set cardinality, both summation and average pooling are necessary; only one would not be sufficient without adding N as a global conditioning feature (which we avoided for the sake of simplicity).

All set- and graph-based layer structures, such as References [176, 178, 277], can be seen as specific cases of the graph network block of Reference [177]. This is also true for the EPiC layers, where we made the following implementation choices:

- No edge features are used to allow for a linear scaling of the computational cost.
- The point attribute transformation ϕ^p is performed after the global transformation ϕ^g to allow for global information aggregation already within one EPiC layer.
- Both summation and average pooling are used in the permutation-invariant aggregation function $\rho^{p\to g}$ as summation as an injective aggregation operator preserves set cardinality information in conjunction with average pooling, which in turn supports faster model convergence as it scales cardinality-independent.

This formulation of the EPiC layers and the EPiC-GAN is, to the best of our knowledge, a novel contribution to the machine learning literature, not just in particle physics. A comparison to the full graph network block and the Deep Sets models is shown in Figure 3.4.

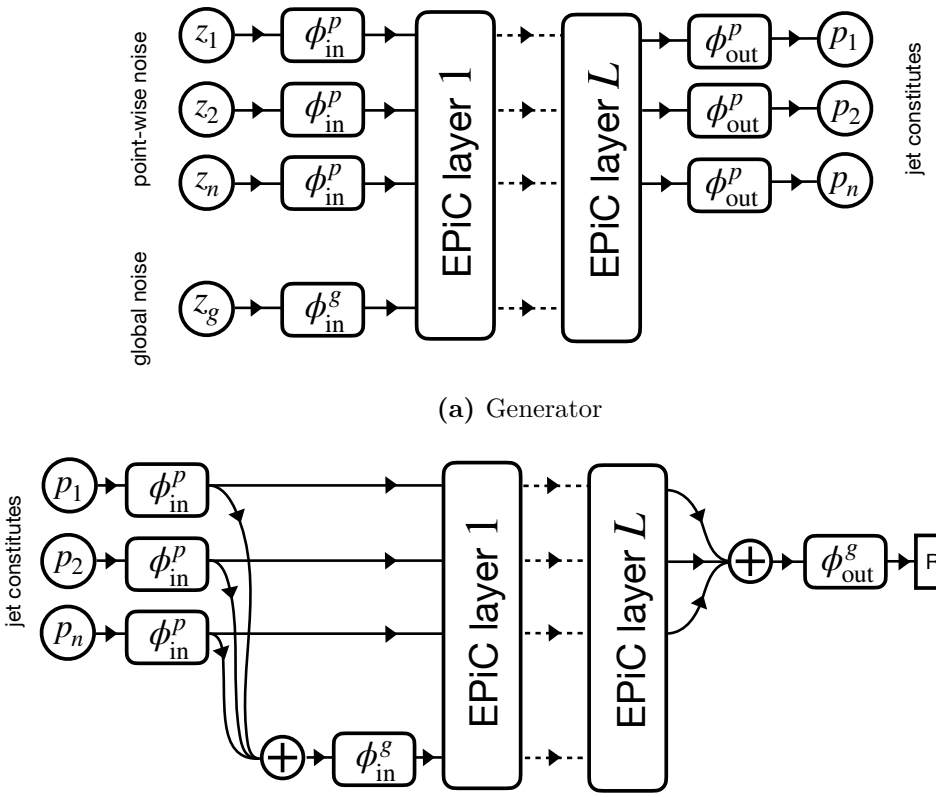
The EPiC layers allow a minimal formulation of a graph network block by limiting the amount of inter-point information transfer. This limitation can be induced by choosing the number of global attributes, i.e. the dimensionality $\dim(\boldsymbol{g})$, as well as by the choice of the number of stacked EPiC layers in the overall model architecture. This "bottlenecking" of the information transferred between points can be optimized to gain a model with the minimum information sharing necessary to perform a given task. Additionally, this limited global latent space allows interpretability of the network in terms of learned global features, as discussed below in Section 7.3.

7.2.2 EPiC-GAN Architecture

The architectures of the generator and discriminator of the EPiC-GAN are shown in Figure 7.3. Both consist of a number of EPiC layers L as well as (permutation-equivariant/ -invariant) blocks ϕ^p_{in}, ϕ^p_{out}, ϕ^g_{in} and ϕ^g_{out} for input- and output dimensionality reduction/expansion of either the global or point attribute vectors, respectively. Residual connections [170] are added to every EPiC layer as well as between the input to the first EPiC layer and the output of every EPiC layer. These residual connections allow the model to learn to "skip" one or multiple EPiC layers. For simplicity, these connections are not shown in Figure 7.3.

Generator

The generator is shown in Figure 7.3a. For jet generation, depending on the jet particle multiplicity N, the particles are randomly initialized as $z_{i\in[1,N]}$ from normal noise $\mathcal{N}(0,1)$, each with three features (same as the output features: $p_{\text{T}}^{\text{rel}}$, η^{rel}, and ϕ^{rel}). Additionally,



(a) Generator

(b) Discriminator

Figure 7.3: Overview of the EPiC-GAN architecture. Generator **(a)** and discriminator **(b)** consist of multiple EPiC layers (see Figure 7.2) as well as weight-shared neural networks for input/output dimensionality expansion/reduction. The $\oplus$ symbol describes the permutation-invariant aggregation function $\rho^{p \to g}$ with both element-wise summation and average pooling. Additional residual connections between EPiC layers are described in the text. Figures originally published in Reference [2].

the global attribute vector z_g is initialized with a fixed dimensionality $\dim(\boldsymbol{g})$ (see all hyperparameter choices in Table 7.1). The dimensionality expansion and reduction layers ϕ^p_{in} and ϕ^p_{out} are learned by a 1-layer MLP with LeakyReLU activations and the global dimensionality expansion block ϕ^g_{in} is learned by a 2-layer MLP with LeakyReLU activations. These random vectors are transformed by $L_{\text{generator}} = 6$ EPiC layers and result in generated jet constituents $p_{i\in[1,N]}$.

Discriminator

The jet constituents $p_{i\in[1,N]}$ are the input to the discriminator (shown in Figure 7.3b). After dimensionality expansion via ϕ^p_{in}, a permutation-invariant aggregation with average and summation pooling is performed to create a global attribute vector $\boldsymbol{g}$. This output is fed through ϕ^g_{in} and then used as input to the first EPiC layer. The discriminator was implemented with $L_{\text{discriminator}} = 3$ EPiC networks. After the final EPiC layer, another average and summation aggregation is performed to create a permutation-invariant feature vector that is fed into the output block ϕ^g_{out}, which has a single Sigmoid output activation to discriminate between real and fake (generated) jets. In the discriminator, ϕ^p_{in}, ϕ^g_{in}, and ϕ^g_{out} are implemented with 2-layer MLPs with LeakyReLU activations.

7.2.3 EPiC-GAN Training

For training the EPiC-GAN, each batch needs to be composed of jets with the same particle multiplicity. As the particle multiplicity is variable (up to 30 for JetNet30 and up to 150 for JetNet150), we implemented a dynamic PyTorch dataloader class, which composes batches of equal cardinality with a maximum batch size as a hyperparameter. In each epoch, the batches are shuffled anew. However, for some cardinalities with fewer jets in the dataset than the maximum batch size (i.e. $N > 10$) the batches remain the same in each epoch. During sampling, the set cardinality for all generated jets is sampled from a kernel density estimation (KDE) of the particle multiplicity distribution of the training set. Like during training, these cardinalities are clustered into batches of the same cardinality when generating novel jets.

An alternative approach to this cardinality clustering strategy was implemented in the EPiC-FM and EPiC-JeDi models (see Section 7.4) (as well as in the CaloClouds models (Section 6.2)). In EPiC-FM the maximum number of points are always sampled/used while zero-padding unphysical points. Masking is applied such that in each pooling operation as well as in the final generation step, the masked points (i.e. the zero-padded points) are set to zero. This has the advantage, that no custom dataloader class needs to be implemented, although it is slightly less computationally efficient.

Due to the particle multiplicity cut at either $N = 30$ or $N = 150$, the jets in the JetNet30 and JetNet150 datasets are not necessarily centered anymore. Therefore, we apply as a preprocessing step a re-centering to $\eta^{\text{jet}} = 0$ and $\phi^{\text{jet}} = 0$ based on the subset of kept particles. This centering is also applied as a postprocessing step to the generated jets. To allow for a consistent evaluation, we also apply this step to both the test events and the MP-GAN-generated events, although the changes are so small that a difference is noticeable neither in the histograms nor in the evaluation scores.

Table 7.1: Hyperparameters used for the EPiC-GAN trainings. Table adapted from Reference [2].

Hyperparameter	Value
$L_{\text{generator}}$ EPiC layers	6
$L_{\text{discriminator}}$ EPiC layers	3
$\dim(\boldsymbol{g})$ global dimensionality	10
$\dim(\boldsymbol{p}_i)$ point dimensionality	3
Hidden dimensionality	128
Activation function	LeakyReLU(0.01)
ADAM [166] learning rate	10^{-4}
Max. batch size	128
Max. training epochs	2,000
Generator weights	$\sim$ 425,000
Discriminator weights	$\sim$ 313,000

As an additional preprocessing step, we standardized the features to follow a normal distribution with $\mathcal{N}(0, 5^2)$. We found that this five times wider standardization than a unit Gaussian improves the discriminator performance since low particle $p_{\text{T}}^{\text{rel}}$ values are numerically more separated than just with a standard unit Gaussian (a similar effect can be achieved by taking the logarithm of the input as preprocessing). To avoid unphysical (negative) particle p_{T} values, we apply a minimum $p_{\text{T}}^{\text{rel}}$ cut, i.e. we set all $p_{\text{T}}^{\text{rel}}$ values below the minimum $p_{\text{T}}^{\text{rel}}$ of the training set to that value.

We train the EPiC-GAN with the Least Squares GAN (LSGAN) [189] objective using the binary 0-1 labeling scheme (see Section 4.1.1). Compared to the vanilla GAN objective [228], which employs binary cross-entropy as a loss function, we found that the least squares objective leads to more stable training since it avoids vanishing gradients. To further stabilize the GAN training, we employ weight normalization [278] in all hidden layers. The hyperparameter choices for the EPiC-GAN are presented in Table 7.1.

The epoch for model evaluation ("best epoch") is chosen based on the generated relative jet mass distribution compared to the one of the validation set ($\sim 25,000$ samples). To make a stable estimation of the generative fidelity of this distribution, we generated 10 sets of jets, each with the same statistics as the validation set. We chose the best model based on the mean of the 1-Wasserstein distance between the validation distribution and each generated distribution. Since GANs are inherently unstable to train, we opted to train the GAN for each dataset three times and present the best model out of the three here. A similar procedure is done in Reference [62] for the training of the MP-GAN and we are therefore able to fairly compare the two generative models. Further, the jet mass is correlated to various other jet physics observables, including numerous Energy Flow Polynomials (EFPs) [131], and is therefore an overall good indicator for a well-performing model.

7.3 Jet Generation with the EPiC-GAN

The EPiC-GAN is applied to the generation of JetNet30 and JetNet150 particle jets. At the time of publishing the EPiC-GAN, the state-of-the-art equivariant generative model for this dataset was the MP-GAN [62], for which weights are publicly available. Hence, we compare the EPiC-GAN generative performance here with the MP-GAN as well as the test dataset.

The MP-GAN, however, was only trained on the JetNet30 datasets, likely because the message-passing layers scale quadratically with the point cloud cardinality resulting in computationally impractical training performance on the larger JetNet150 dataset. Hence, with the EPiC-GAN we introduce the first generative model trained on the JetNet150 dataset.

We further limit the discussion in this section to the JetNet30 and JetNet150 top datasets. For the top decay, we consider its hadronic decay channel via the b quark and the W boson with the decay chain $t \to bW \to bqq'$ into three quarks. Since with $R = 0.4$ a rather small anti-k_T clustering radius was chosen, not all three quarks are always contained within the jet. This leads to two distinct peaks in the jet mass distributions: the main peak at the t quark mass and a secondary peak at the W boson mass. Therefore, the phase space of the top dataset is more complex than the one for the gluon and light quark decays, which is why we limit our discussion here to the top. Arguably the generation of jets with a clustering radius of $R = 0.8$ and fully contained top jets would physically be more practical, however, to benchmark a generative machine learning model, this complex structure is more challenging and offers a more interesting playground. Our conclusions apply equally to the gluon and light quark datasets and results for these can be found in Reference [2].

Overall we shall observe that the EPiC-GAN reaches a comparable fidelity as the MP-GAN, which is a very satisfying result, since it is computationally significantly more efficient and offers the opportunity to work on larger point cloud datasets such as JetNet150.

7.3.1 JetNet30 Top Generation

To investigate the physics performance of the EPiC-GAN on the JetNet top datasets, we generated the same amount of jets as in the test dataset ($\sim 25,000$) and calculated various physics observables. In this section, we qualitatively assess the performance of the generative models using differential distributions of nine observables: the relative particle features $p_\mathrm{T}^\mathrm{rel}$, η^rel, and ϕ^rel (aggregated over all particles of all jets), the 1^st, 5^th, and 20^th leading relative particle $p_\mathrm{T}^\mathrm{rel}$ per jet, the particle multiplicity N, the relative jet mass $m_\mathrm{jet}^\mathrm{rel}$, and the relative jet $p_\mathrm{T,jet}^\mathrm{rel}$.

We show the resulting distributions for the test set, the EPiC-GAN, and the MP-GAN generated jets in Figure 7.4. The relative particle $p_\mathrm{T}^\mathrm{rel}$ is very well reproduced by both GANs. The same is true for the particle η^rel distributions, although both GANs do not generate outliers $|\eta^\mathrm{rel}| > 0.4$ like the ones present in the test set. The particle ϕ^rel distribution is similarly well generated, although the MP-GAN creates a distribution slightly too broad. Looking in detail at the leading relative $p_\mathrm{T}^\mathrm{rel}$ distributions, we observe comparably good performance for both GANs in the first and fifth leading particle $p_\mathrm{T}^\mathrm{rel}$, yet the EPiC-GAN achieves a slightly better representation of the high and low p_T tail of the 20^th particle. For the jet observables, we first explore the particle multiplicity distribution, which is similarly

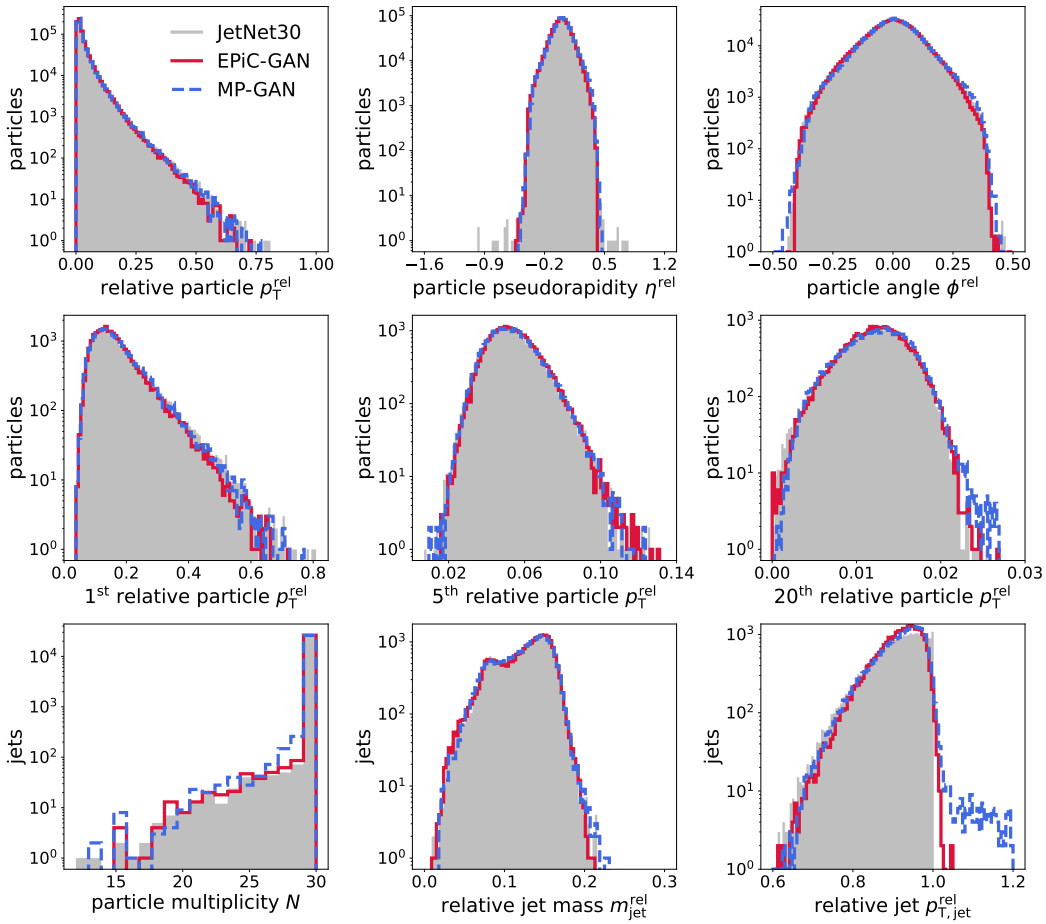


Figure 7.4: JetNet 30 top dataset: Various particle- and jet-level differential distributions comparing the test set, MP-GAN generated, and EPiC-GAN generated jets. Figures originally published in Reference [2].

Table 7.2: Wasserstein-based evaluation scores and Fréchet ParticleNet distance (FPND) for the JetNet 30 top dataset. The "truth" values are calculated between the test and training set. The MP-GAN scores are calculated using the trained models from Reference [62]. Table adapted from Reference [2].

Jet class	Model	W_1^{M} ($\times 10^{-3}$)	W_1^{P} ($\times 10^{-3}$)	W_1^{EFP} ($\times 10^{-5}$)	FPND
Top	Truth	0.2 ± 0.1	0.3 ± 0.1	0.6 ± 0.5	0.02 ± 0.01
	MP-GAN	$\mathbf{0.5 \pm 0.1}$	2.4 ± 0.2	$\mathbf{1.0 \pm 0.7}$	0.35 ± 0.04
	EPiC-GAN	$\mathbf{0.5 \pm 0.1}$	$\mathbf{2.1 \pm 0.1}$	1.7 ± 0.3	$\mathbf{0.31 \pm 0.03}$

well modeled by both GANs — although in the case of the EPiC-GAN, this is actually due to the cardinality KDE. As one of the most challenging distributions, the relative jet mass $m_{\mathrm{jet}}^{\mathrm{rel}}$ is well represented by both GANs with the double peak feature due to the W and top mass being distinctly visible. Finally, the relative jet $p_{\mathrm{T,jet}}^{\mathrm{rel}} < 0.9$ is well generated by both GANs. Above 0.9 both GANs deviate from the test distribution. The MP-GAN generates large outliers, while the EPiC-GAN generates maximum values below 1.05. This cut-off at $p_{\mathrm{T,jet}}^{\mathrm{rel}} = 1.0$ is due to the normalization of the relative jet p_{T} and could be introduced with the GANs as a post-processing calibration. Yet, without calibration, it is interesting how well the models can generate the cut-off. Here, we see an advantage of the EPiC-GAN over the MP-GAN.

Overall we observe a comparable generative fidelity between the EPiC-GAN and the MP-GAN in this qualitative assessment of differential distributions for the JetNet30 top dataset. Next, we approach this comparison quantitatively with evaluation scores based on the 1-Wasserstein distance between physical distributions.

For these evaluation scores, we calculate similar ones as in Reference [62]. These scores are either based on the 1-Wasserstein distance between certain physics distributions or based on the Fréchet distance between latent space of a multi-classifier. For the Wasserstein scores, we use the same observables as in Reference [62]: the Wasserstein distance between the relative jet mass distribution W_1^{M}, the mean of the Wasserstein distance between the relative particle feature distributions ($p_{\mathrm{T}}^{\mathrm{rel}}$, η^{rel}, and ϕ^{rel}) W_1^{P}, and the mean of Wasserstein distance between five Energy Flow Polynomial (EFP) distributions (the loop-less multi-graphs with 4 nodes and 4 edges) W_1^{EFP}. We further calculate the Fréchet ParticleNet distance (FPND) using trained ParticleNet weights implemented in the JetNet package [62].

Compared to Reference [62], we increase the statistics for calculating the $W_1^{\mathcal{O}}$ (for observable $\mathcal{O}$) and FPND scores. With the "truth" scores, we compare the training set with the test set to estimate the score an "optimal" generative model would achieve. We calculate four "truth" Wasserstein scores between the test set and four subsets of the training data with equal size and quote the mean and standard deviations of the four. For the "truth" FPND, we quote similarly the mean and standard deviation of four scores using the same subsets. The MP-GAN and EPiC-GAN scores are calculated using ten subsets (i.e. we generated about $\sim 250,000$ jets with each model) and we report their mean and standard deviation.

Table 7.3: Wasserstein-based evaluation scores and FPND for the JetNet 150 top dataset. The "truth" values are calculated between the test and training set. Table adapted from Reference [2].

Jet class	Model	W_1^{M} ($\mathrm{x}10^{-3}$)	W_1^{P} ($\mathrm{x}10^{-3}$)	W_1^{EFP} ($\mathrm{x}10^{-5}$)
Top	Truth	0.3 ± 0.1	0.2 ± 0.1	1.3 ± 0.8
	EPiC-GAN	0.6 ± 0.1	3.7 ± 0.3	2.8 ± 0.7

In Table 7.2 we present the resulting scores for the MP-GAN and the EPiC-GAN compared to the truth values. As already qualitatively observed, both generative models perform very similarly. Considering the standard deviations, all scores are in the margin of error from each other. Comparing the GAN scores to the Monte-Carlo truth, we find that the GAN samples are still distinguishable from the truth samples, which is also apparent from the histograms. In particular, the particle-level features calculated with W_1^{P} and the FPND are quite off. However, the scale of the FPND is difficult to interpret as it is unclear what features the ParticleNet classifier is sensitive to. Note that in Reference [62] two more scores were calculated — coverage and minimum matching distance — however, it was observed that these scores were not sensitive enough to be useful for the evaluation of the generative models.

Overall, we conclude that both GANs perform equally well on the task of generating JetNet30 top jets. Next, we evaluate the performance of the EPiC-GAN on the JetNet150 top dataset, for which the EPiC-GAN constitutes the first generative model published.

7.3.2 JetNet150 Top Generation

We trained the EPiC-GAN on the JetNet150 top dataset with the same hyperparameters as for the JetNet30 top dataset. Here, we compare the EPiC-GAN only to the test set, since the MP-GAN is not available for JetNet150. The same nine distributions as in the previous section are shown in Figure 7.5.

The overall relative particle features, $p_{\mathrm{T}}^{\mathrm{rel}}$, η^{rel}, and ϕ^{rel}, are accurately modeled by the EPiC-GAN, although the high angle tail of the ϕ^{rel} distribution is slightly underestimated. The leading relative $p_{\mathrm{T}}^{\mathrm{rel}}$ distributions are also well modeled, except for the 20^{th} leading particle, where the distribution is slightly shifted to higher values. As a jet observable, the particle multiplicity is very well reproduced by the KDE of the EPiC-GAN. The relative jet mass distribution is also very well modeled. This was also the distribution for discriminating the best epoch choice. Finally, the relative jet $p_{\mathrm{T,jet}}^{\mathrm{rel}}$ is narrowly distributed around 1.0, yet the mean is slightly above 1.0. As previously mentioned, this could be calibrated with a post-processing step, however, it is a good benchmark that the EPiC-GAN can generate this cut-off reasonably well without any calibration.

For the quantitative evaluation scores, we calculate the same Wasserstein-based scores as for the JetNet30 top dataset. We do not include the FPND here, since its calculation for JetNet150 was not supported by the JetNet package. The calculated mean and standard

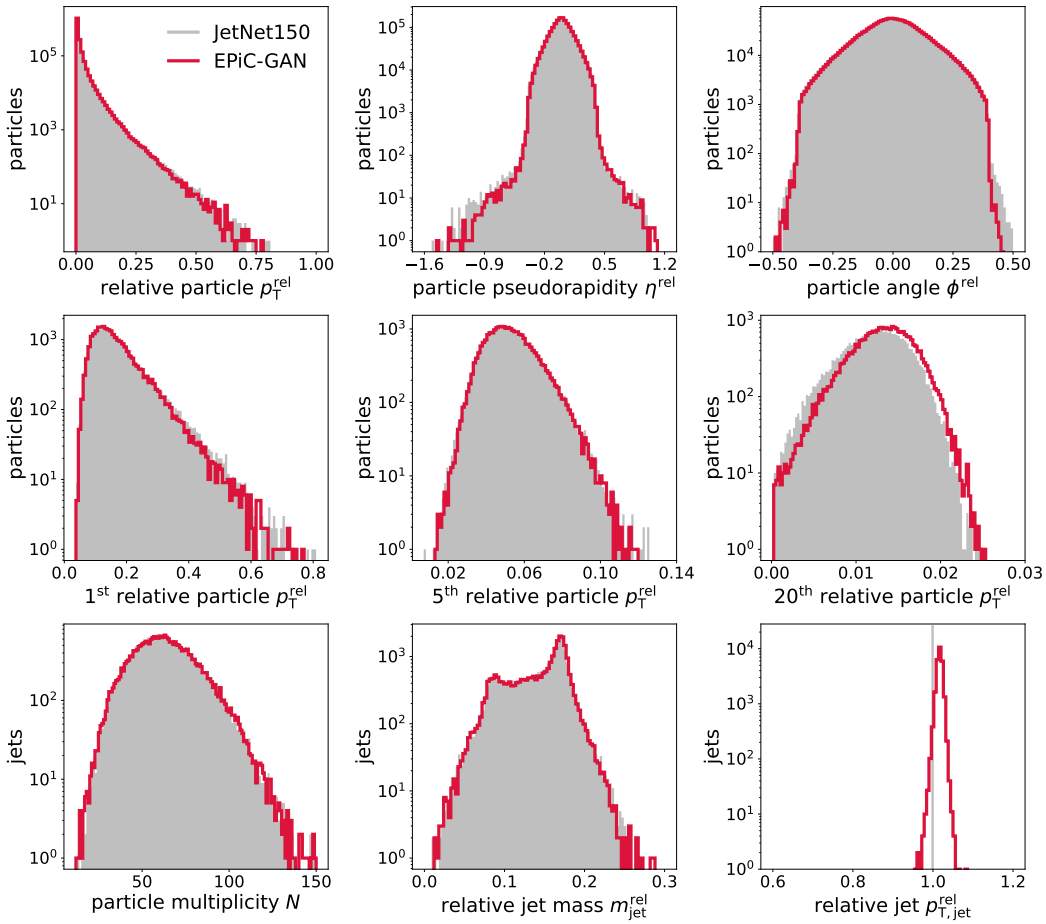


Figure 7.5: JetNet 150 top dataset: Various particle- and jet-level differential distributions comparing the test set and EPiC-GAN generated jets. Figures originally published in Reference [2].

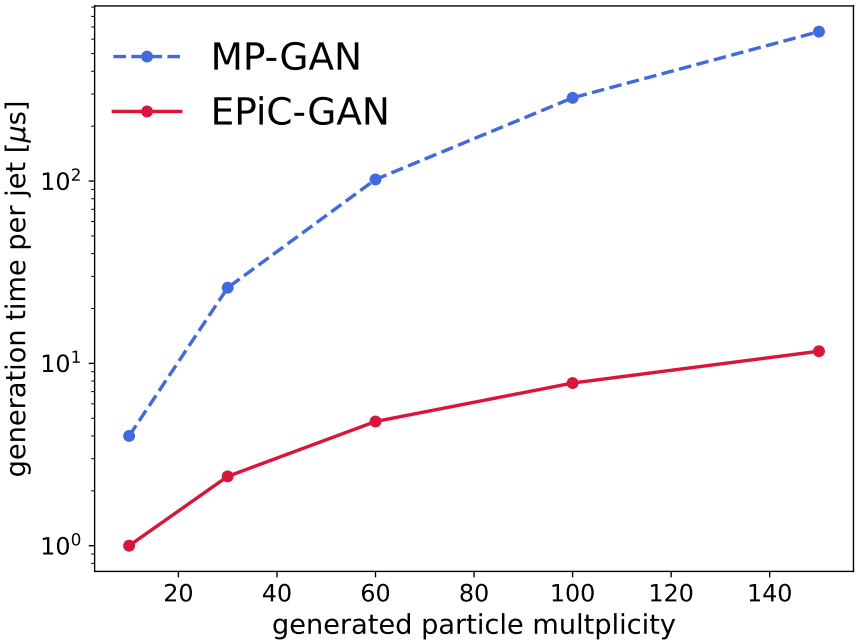


Figure 7.6: Timing performance of the EPiC-GAN compared to the MP-GAN as a function of a (fixed) generated particle multiplicity. For either GAN, a total of 500k jets were generated. The batch size was optimized for optimal generation speed and generation was performed on a system with a single NVIDIA® A100-40GB GPU. Figure originally published in Reference [2].

deviation of the calculated scores for the test set compared to either the training set ("truth") or to the EPiC-GAN generated samples are shown in Table 7.3. Comparing the EPiC-GAN scores to the truth scores, we find that the EPiC-GAN samples are worse than the truth samples, in particular for the particle-level features. However, the scores are only slightly worse than the ones calculated on the JetNet30 top dataset.

Overall, we observe that the EPiC-GAN performs comparably well on the JetNet150 top dataset as on the JetNet30 top dataset. Yet, more research is needed to achieve truth-level generative fidelity. While the generative performance of the EPiC-GAN is in line with the MP-GAN, it is computationally significantly more efficient. Its timing performance is discussed in the next section.

7.3.3 Timing

As the EPiC layers scale linearly with the cardinality of the point cloud and the fully connected message-passing layers scale quadratically, the EPiC-GAN is computationally more efficient and allows for a faster generation of larger point clouds than the MP-GAN on the same hardware. In Figure 7.6 we show the generation time per jet for a fixed particle multiplicity for both the EPiC-GAN and the MP-GAN. We ran both GANs on the same hardware — a single NVIDIA® A100-40GB GPU — and generated 500k jets with each model. For both models, we used randomly initialized weights, since a trained model is not necessary for this

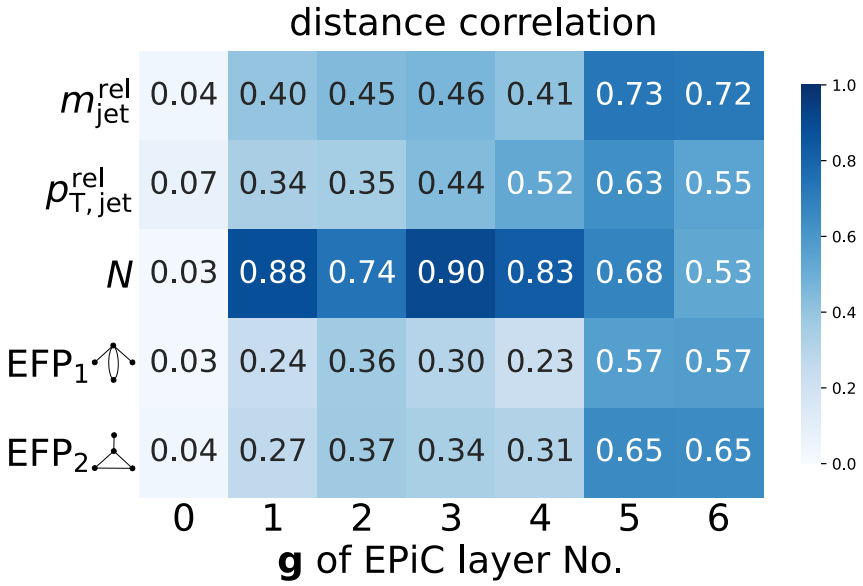


Figure 7.7: Distance correlation between several physical jet observables and the global attribute vector $\boldsymbol{g}$ after (and before) each EPiC layer. The JetNet 30 light quark training is shown. Figure originally published in Reference [2].

comparison.

The EPiC-GAN generates a jet with 30 constituents 13x faster than the MP-GAN (2 μs vs. 26 μs), while for 150 particles its 55x faster (12 μs vs. 660 μs). Note that these are the generation times for a fixed particle multiplicity. JetNet30 and JetNet150 contain jets with a variable cardinality and therefore the average generation time is lower than for example the quoted time for fixed 30 particles. This better scaling behavior of the EPiC-GAN allows it to be used for the generation of larger point clouds such as the JetNet150 dataset and even larger point clouds such as calorimeter point clouds. For example, we applied the EPiC layers in the encoder of the CaloClouds model to calorimeter point clouds with a cardinality of up to 6,000 (see Chapter 6).

7.3.4 Interpretability

Several global variables, such as the jet mass, the jet p_{T}, and the particle multiplicity, define the overall particle jet. Ideally, the global attribute vector $\boldsymbol{g}$ of the EPiC layer should learn such global features. To study, if and what kind of physical information is learned/encoded in the multi-dimensional global attribute vector, we calculate several important jet variables of 5,000 EPiC-GAN generated jets and calculate their distance correlation [279] to the encoded $\boldsymbol{g}$ space of the generator. These variables include the relative jet mass $m_{\text{jet}}^{\text{rel}}$, the relative jet $p_{\text{T,jet}}^{\text{rel}}$, the particle multiplicity N, and two energy flow polynomials (EFP) — two out of the five EFPs calculated for the W_1^{EFP}.

The resulting distance correlations are shown in Figure 7.7. We show the distance correlation between the observables and the vector $\boldsymbol{g}$ before the first EPiC layer ($\boldsymbol{g}$ of EPiC

layer No. 0) and after each of the six EPiC layers ($\boldsymbol{g}$ of EPiC layer No. 1-6). We observe virtually no correlations before the first EPiC layer, but already strong correlations with the particle multiplicity after the first EPiC layer. Correlations with all other variables are increasing with the number of EPiC layers. Notably, the strongest correlation (0.90) with the particle multiplicity is observed after the third EPiC layer. The second-strongest correlation (0.73) is with the jet mass after the fifth EPiC layer, which is particularly well modeled by the EPiC-GAN, as this is the variable used for the final evaluation epoch choice. This indicates, that indeed these physical observables are learned by the EPiC-GAN.

Observing these distance correlations with the global attribute space allows one to monitor the training and interpret the physics learned by the EPiC model. Depending on the application, this might increase the trust in a model based on EPiC layers, if one observes that important physical features are learned by the model.

7.4 EPiC Diffusion Models for Particle Jets

With the high fidelity that can be reached using the EPiC layers in the EPiC-GAN model, as well as the advantages of these layers in terms of computational efficiency, scaling, and interpretability, we endeavor to use them together with more recent advancements in generative modeling: diffusion and flow matching generative models.

For generative modeling tasks in computer vision, diffusion generative models [213–215, 217, 280] — often also referred to as score-based generative models — have largely surpassed the fidelity of GANs, albeit at an increased computational cost [216]. Recently, the flow matching [210–212] paradigm for training continuous normalizing flows (CNFs) [207] has been introduced, which makes training of CNFs significantly more stable and lets them approach or even surpass the fidelity of some diffusion model implementations. An introduction to CNFs is given in Section 4.3.2 and an overview of diffusion, score matching and flow matching is given in Section 4.4.

At the time of publishing the results of the studies presented in this section, diffusion models were used in high-energy physics for the generation of particle jets [258, 262, 263, 281] and calorimeter showers as voxelized images [59–61, 282]. Flow matching was used for generating jet-level observables [281]. Further examples of the application of diffusion models for calorimeter shower generation can be found in Chapter 6.

Shortly after the publications of the EPiC-GAN results in Reference [2], the PC-JeDi (Point Cloud Jets with Diffusion) [262] model was introduced, which achieves comparable performance to the EPiC-GAN and even surpassing it in certain sub-jet observables. However, as a transformer-based diffusion model, which scales quadratically with the particle multiplicity and requires 200 model passes for denoising, this increase in fidelity comes at a significant computational cost. This computational cost further makes PC-JeDi impractical for generating point clouds larger than the ones in the JetNet30 dataset, such as JetNet150.

Together with the authors of PC-JeDi, we implemented an EPiC-variant of the PC-JeDi model, dubbed EPiC-JeDi, using a similar layer structure as the EPiC-GAN, but with the PC-JeDi diffusion model parameterization. Further, we trained the same model with the optimal-transport flow matching objective from Reference [212] to compare the two

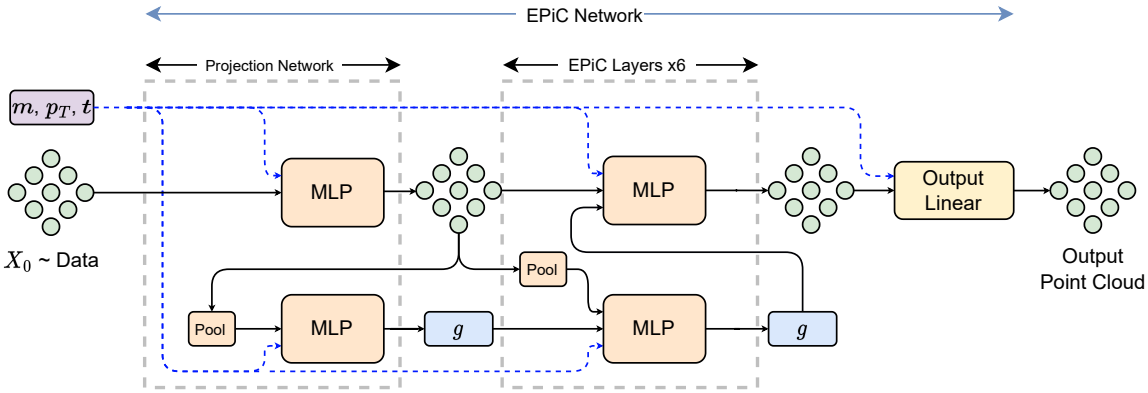


Figure 7.8: Overview of the "EPiC Network" architecture used in both EPiC-JeDi and EPiC-FM. It is based on the EPiC-GAN generator (see Figure 7.3). Each multi-layer perceptron (MLP) is a two-layer neural network and the pooling operations are summation and averaging. The unconditional models only use the time step t as conditioning. Figure originally published in Reference [4].

training objectives. This model we call EPiC-FM. However, during inference, both models are identical (except their trained weights) as they both utilize the midpoint ODE solver. Since no stochasticity is involved in the sampling process, both models can be technically referred to as CNFs.

We observe, that both approaches outperform PC-JeDi and EPiC-GAN and that EPiC-FM even outperforms EPiC-JeDi, suggesting an advantage for the flow matching objective. However, compared to EPiC-GAN, these performance increases come at a significantly higher computational cost.

7.4.1 EPiC-JeDi and EPiC-FM Architecture

EPiC-JeDi and EPiC-FM share the same model architecture and only deviate in their training objective. During sampling, they resemble a CNF, we will therefore refer to both models as EPiC-CNF whenever a technical detail applies to both models. The models presented here are designed to work for both the JetNet30 and the JetNet150 datasets.

The network architecture of EPiC-CNF is dubbed *EPiC Network*. An overview of the EPiC Network is shown in Figure 7.8. Similar to most diffusion model architectures — usually designed as U-Nets [171] — the input and output dimensionality must be equal. Six consecutive EPiC layers make up the bulk of the EPiC Network. Each MLP in the network is implemented with a hidden dimensionality of 128 and LeakyReLU activations. The input point cloud has a feature dimensionality of three, corresponding to the relative particle $p_{\mathrm{T}}^{\mathrm{rel}}$, relative pseudorapidity η^{rel}, and the relative azimuthal angle ϕ^{rel} of the jet constituents. To expand the input dimensionality to the hidden network dimensionality and to create the global attribute vector $\boldsymbol{g}$ (with $\dim(\boldsymbol{g}) = 10$), a *Projection Network* is set before the first EPiC layer. This Projection Network is made up of two MLPs and the same average and summation pooling as in the EPiC layer. A single linear output layer without activation is

used for dimensionality reduction after the sixth EPiC layer.

Two changes are made to the EPiC layers in EPiC-CNF in comparison to the ones used for the EPiC-GAN: the addition of conditioning and masking of zero-padded particles in the pooling layers. The conditioning is needed as diffusion models are conditioned on the time step t. Additionally, this conditioning can be used to steer the model with additional jet properties such as the jet mass or the jet p_{T}. The conditioning is implemented as a concatenation to the input of every layer of both MLPs (ϕ^p and ϕ^g in Figure 7.3). This conditioning also applies to the Projection Network and the output layer. For the studies in Reference [4], we have implemented two versions of the EPiC-CNF models: an unconditional model (only conditioned on the time step t, which is mandatory for diffusion models) and a conditional version (adding jet mass and jet p_{T} conditioning; the same conditioning used for PC-JeDi [262]). As the unconditional version can be directly compared to the (unconditional) EPiC-GAN, we will focus in this section on the unconditional EPiC-CNF models. Detailed comparisons of the conditional EPiC-CNF models with the (conditional) PC-JeDi model can be found in Reference [4].

Masking is added to the EPiC layers via a separate input for the mask which represents whether a particle actually should be generated or is a zero-padded particle (to allow for jets with variable cardinality within a single batch). As particles communicate only via the global function ϕ^g, the masked particles are set to zero before the permutation-invariant aggregation. This way, ϕ^g is applied only to non-padded (existing) particles. Note that the local function ϕ^p is applied to both masked (zero-padded) and unmasked (existing) particles, but since they do not contribute to the global attribute vector $\boldsymbol{g}$, this does not change the result. In the generated output particle cloud, the masked particles are set to zero in the same way and disregarded in downstream jet observable calculations. This masking is mainly done for convenience as it allows the omission of a preprocessing step necessary for the EPiC-GAN, in which the jets were sorted in equal cardinality batches.

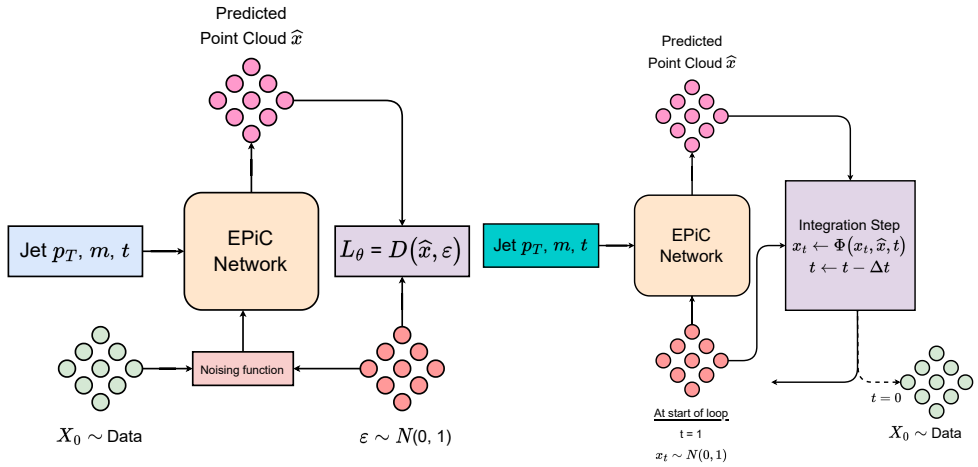
7.4.2 Training and Sampling

Here, we discuss the specific implementation choices that were made for EPiC-JeDi as a diffusion model and for EPiC-FM as a CNF trained with flow matching. An overview of the training and sampling process of both models is shown in Figure 7.9.

The EPiC-JeDi training objective is the same for PC-JeDi outlined in Reference [262]. It is inspired by score-based generative modeling through SDEs (see Section 4.4.2) and follows the variance preserving framework discussed in Reference [215] and the exact implementation follows Reference [266]. The overall loss function for training the score model $\mathbf{s}_{\boldsymbol{\theta}}$ in EPiC-JeDi is given by

$$L_{\mathrm{JeDi}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\boldsymbol{\epsilon},\mathbf{x}_t}\left[\left(1+\alpha\frac{\beta(t)}{\sigma(t)^2}\right)\left\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,y,t)-\boldsymbol{\epsilon}\right\|^2\right], \tag{7.3}$$

where α is a weighting parameter set to $\alpha = 10^{-4}$, $\beta(t)$ and $\sigma(t)$ are variance schedulers, and y denotes conditional information for the conditional version of EPiC-JeDi. The expectation is taken over the time step t uniformly sampled from $\mathcal{U}(0,1)$, the total noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, and the noised input $\mathbf{x}_t = \gamma(t)\mathbf{x}_0 + \sigma(t)\epsilon$ where $\mathbf{x}_0 \sim p_{\mathrm{data}}$ is the input point cloud and $\gamma(t) = 1-\sigma(t)^2$.



(a) Training **(b)** Sampling

Figure 7.9: Pipeline of the EPiC-JeDi and EPiC-FM training **(a)** and sampling **(b)** process. Figure originally published in Reference [4].

EPiC-FM on the other hand is trained with the optimal-transport conditional flow matching objective from Reference [212]. A derivation of flow matching and its implementation via conditional flow matching is given in Section 4.4.3. The EPiC-FM loss function is given by Equation 4.74 and the minimum noise is set to $\sigma_{\min} = 10^{-4}$.

The two training objectives differ in two main aspects: (1) The JeDi loss function trains a model that essentially predicts the noise added to the input data and the flow matching loss function trains a model that predicts the perturbed data directly. (2) The linear noise-to-data scaling in optimal-transport flow matching leads to straight probability trajectories, while the variance-preserving framework in score-based modeling exhibits curved probability paths. From this perspective, the flow matching objective may be advantageous for efficient training and sampling with fewer model evaluations [212].

The EPiC-CNF models are implemented with PyTorch and we used the Adam-W optimizer [283] with an initial learning rate of 10^{-3}. A cosine learning rate scheduler is employed, including warm-up for 1,000 epochs. Both models were trained[6] for 10,000 epochs with a batch size of 1,024 and the last epoch was chosen for model evaluation. For the differential equation solvers, we used the torchdiffeq library [284]. Further hyperparameters as well as the model and training data sizes can be found in Table 7.4.

Both EPiC-CNF models are sampled with the same midpoint ODE solver, a 2^{nd}-order Runge-Kutta variant. We found this solver works best for both models and we use it with a fixed 200 number of function evaluations (NFE), i.e. with 100 solver steps. This is the same number of solving steps used with the DDIM and Euler-Maruyama (EM) [222] sampler in PC-JeDi. Further results on experiments with other solvers for the EPiC-CNF models can be found in the appendix of Reference [4].

To sample the correct cardinality (particle multiplicity) for the initial point cloud, we use

[6]The presented EPiC-FM model was trained by Cedric Ewen and the presented EPiC-JeDi model was trained by Debajyoti Sengupta.

Table 7.4: Hyperparameters used for the EPiC-JeDi and EPiC-FM trainings. Table originally published in Reference [4].

Hyperparameter	Value
EPiC layers	6
EPiC global dimensionality	10
Hidden dimensionality	128
Activation function	LeakyReLU(0.01)
ADAM-W [283] learning rate	10^{-3}
Learning rate scheduling	Cosine with warm-up
Warm-up epochs	1,000
Batch size	1,024
Training epochs	10,000
Model weights	$\sim 560,000$
Training events	$\sim 110,000$
Test events	$\sim 27,000$

a (discrete) normalizing flow trained on the particle multiplicity distribution — in addition to the jet mass and the jet p_T, which are used only for the conditional EPiC-CNF models. The normalizing flow is implemented with four rational quadratic spline (RQS) coupling blocks [255] in conjunction with invertible linear layers using the NFLOWS [256] library. For training the flow model, we apply de-quantization to the discrete particle multiplicity by adding continuous noise [285] and use during sampling the nearest integer for the jet generation.

7.5 EPiC Jet Generation with Diffusion Models

To study and benchmark the fidelity that can be achieved with the EPiC-CNF models, they were trained on the JetNet30 and the JetNet150 datasets. In the following Section 7.5.1, we compare the unconditional EPiC-CNF models to the EPiC-GAN based on various particle- and jet-level observables. We calculate various evaluation scores for all conditional and unconditional models and show them for both JetNet30 and JetNet150 in Section 7.5.2. A more detailed comparison of the conditional EPiC-CNF models with the PC-JeDi model can be found in Reference [4]. A timing comparison of the EPiC-CNF models to PC-JeDi is provided in Section 7.5.3.

7.5.1 Unconditional Top Jet Generation

To assess the physics performance of the unconditional EPiC-JeDi and EPiC-FM models compared to the EPiC-GAN, we generate 270,000 jets with each model. For comparison, the test set consists of about 27,000 jets. On the particle level, we compare the differential distributions of the 1^{st}, the 5^{th}, and the 20^{th} leading relative particle p_T^{rel} (for the jet

constituents sorted by p_T after generation). As we have seen for the EPiC-GAN, only with such a detailed look at individual constituent distributions we can observe any differences for generated samples on a particle level. To compare the jets on jet level, we calculate the following jet observables: The relative jet mass $m_{\text{jet}}^{\text{rel}}$, the relative jet transverse momentum $p_{\text{T,jet}}^{\text{rel}}$, the relative N-subjettiness ratios τ_{21}^{rel} and τ_{32}^{rel}, and the energy correlator D_2. These observables are in line with Reference [262].

JetNet30 Top Dataset

For the JetNet30 top dataset, the resulting differential distributions for these observables are shown in Figure 7.10. All these distributions include overflow bins and an error band based on the statistical uncertainty of the bin count (which is lower for the test dataset (MC) than for the generated data due to 10× more generated data).

The leading particle p_T^{rel} distribution is well modeled by both EPiC-JeDi and EPiC-FM (within the uncertainty). In comparison, the EPiC-GAN models well the core of the distributions but underestimates the tails. The 5^{th} leading particle p_T^{rel} is particularly well modeled by the EPiC-FM model, i.e. on par with the test distribution. The EPiC-JeDi model slightly overestimates values below the mean, while it underestimates values above the mean. The EPiC-GAN performs similarly, although here it is vice versa. A similar behavior is observable for the 20^{th} p_T^{rel}: Within the uncertainty, the EPiC-FM model is comparable to the Monte Carlo simulation, while both EPiC-JeDi and EPiC-GAN mismodel the low p_T tail.

As previously mentioned, the double-peak relative jet mass distribution of the top dataset is a particularly good distribution for benchmarking the quality of a generative model. This distribution is very well reproduced by both the EPiC-FM and EPiC-GAN models. However, one should note that this distribution was used to make the epoch choice for the EPiC-GAN model, while for the EPiC-FM model simply the last epoch was used for evaluation. The EPiC-JeDi model mismodels the W peak by about 50% and the high mass tail by almost 100%. Similarly, the relative p_T^{rel} is very accurately modeled by EPiC-FM. EPiC-JeDi underestimates the distribution mean and leads to an increased width, while EPiC-GAN oversamples in the mean of the distribution while undersampling the low p_T^{rel} tail. All models produce a few events above the $p_T^{\text{rel}} = 1.0$ maximum of the Monte Carlo distribution.

Jet substructure observables such as N-subjettiness ratios are precision jet features that are difficult to model for a point cloud generative model. The ratio τ_{21}^{rel} is equally accurately modeled by both EPiC-JeDi and EPiC-FM— though both do not exactly reproduce the test distribution. This ratio is slightly worse reproduced by the EPiC-GAN. Larger differences are observable for the τ_{32}^{rel} ratio: both EPiC-JeDi and EPiC-GAN oversample the edges of the distribution while undersampling the center. EPiC-FM reproduces the shape of the distribution much better. The energy correlator D_2 distribution is very well modeled by both EPiC-JeDi and the EPiC-FM model. However, the EPiC-GAN leads to deviations of up to 50%.

Overall we observe the highest jet generation fidelity with the EPiC-FM model. In most distributions, the EPiC-JeDi model is more accurate than the EPiC-GAN.

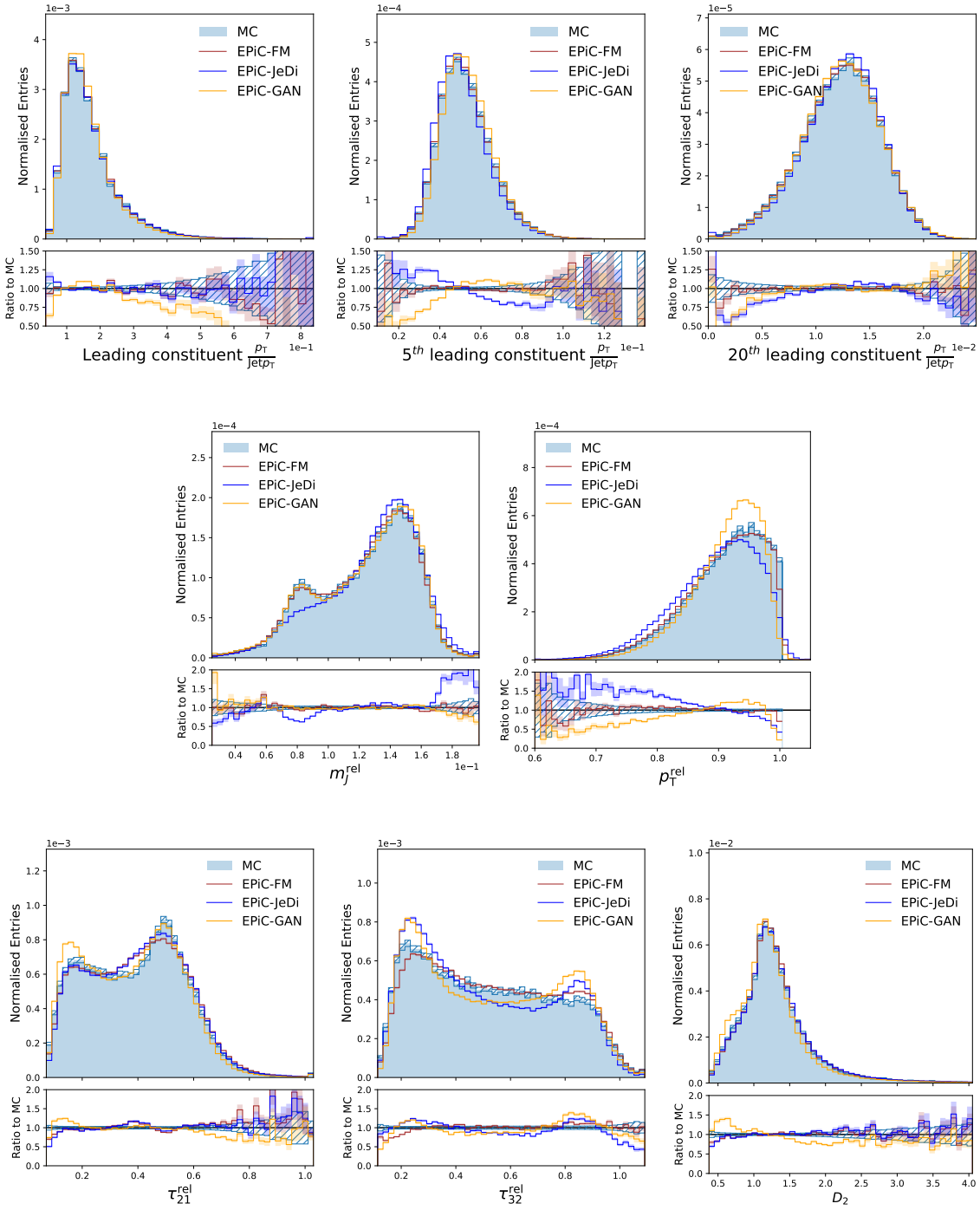


Figure 7.10: Unconditional top jet generation with 30 constituents: Various particle- and jet-level observables for the test set, EPiC-FM, EPiC-JeDi, and EPiC-GAN. **Top row:** relative p_T distribution of the leading (left), 5$^{\text{th}}$ leading (middle), and 20$^{\text{th}}$ leading (right) constituents. **Middle row:** relative jet mass (left) and jet p_T (right) distributions. **Bottom row:** relative jet τ_{21} (left), τ_{32} (middle), and D_2 (right) distributions. The error bands correspond to the statistical uncertainty of the bin count. Figures originally published in Reference [4].

JetNet150 Top Dataset

For the JetNet150 top dataset, the resulting differential distributions for the same observables are shown in Figure 7.11. The leading jet constituent is very well modeled by both EPiC-FM and EPiC-JeDi. Like in the JetNet30 distribution, the EPiC-GAN underestimates the high p_{T} tail of the distribution. The behavior observed in the JetNet30 5^{th} leading constituent distribution is less pronounced for JetNet150: EPiC-FM is very accurate, while EPiC-JeDi and EPiC-GAN slightly mismodel the center and the low p_{T} tail. The 20^{th} leading p_{T} is very well reproduced by both EPiC-FM and EPiC-JeDi. However, strong deviations above 50% are observed for the EPiC-GAN, since it creates a distribution with the mean shifted to higher values.

The relative jet mass distribution is much better modeled by the EPiC-GAN than by either EPiC-FM and EPiC-JeDi. The top mass peak is equally well modeled by the two EPiC-CNF models, yet EPiC-JeDi undersamples the W mass peak (just like for JetNet30). As mentioned before this distribution was used to choose the best epoch of the EPiC-GAN, which likely explains its superior performance in this observable. A stark difference between the models is observable in the relative jet $p_{\mathrm{T}}^{\mathrm{rel}}$ distribution: Due to the particle p_{T} normalization and the high multiplicity cut at 150 particles, almost all jets in the training and test distributions have a $p_{\mathrm{T}}^{\mathrm{rel}}$ of 1.0. This is comparatively well reproduced by the EPiC-FM model. The EPiC-GAN overestimates the $p_{\mathrm{T}}^{\mathrm{rel}}$ and the EPiC-JeDi model underestimates it. Further, both generated distributions with a much larger width than EPiC-FM.

The distribution of the ratio τ_{21}^{rel} is best modeled by the EPiC-GAN, likely due to its correlation with the jet mass. The two EPiC-CNF models perform similarly to each other, but slightly worse than the EPiC-GAN. In the ratio τ_{32}^{rel}, all three models lead to rather large deviations from the test distribution, though overall EPiC-FM is closest. The distribution of energy correlator D_2 is equally well reproduced by all three models.

Overall on JetNet150, we can draw the same conclusions as for the JetNet30 top dataset: Out of the three models, the highest jet generation fidelity is achieved by EPiC-FM.

7.5.2 Evaluation Scores

To quantify the performance of the EPiC-CNF models in more detail, we calculate various evaluation scores based on several particle- and jet-level observables. To this end, we are calculating scores based on the Kullback-Leibler divergence (KLD) for several observables $\mathcal{O}$. We refer to these scores as $\mathrm{KL}^{\mathcal{O}}$ scores.

Since we are here benchmarking very accurate generative models, we replace the previously calculated $W_1^{\mathcal{O}}$ scores with $\mathrm{KL}^{\mathcal{O}}$ scores. The Wasserstein distance represents the minimal "work" that is required to move one probability density distribution onto another. It is particularly well suited for comparing distributions whose supports do not completely overlap. However, for largely overlapping distributions it is less sensitive to density variations than the KLD.

A downside of the KLD in comparison the the Wasserstein distance, is that it requires binning of the distributions, while the Wasserstein distance is an unbinned estimator. This introduces additional hyperparameter choices which previously was a reason to simply use

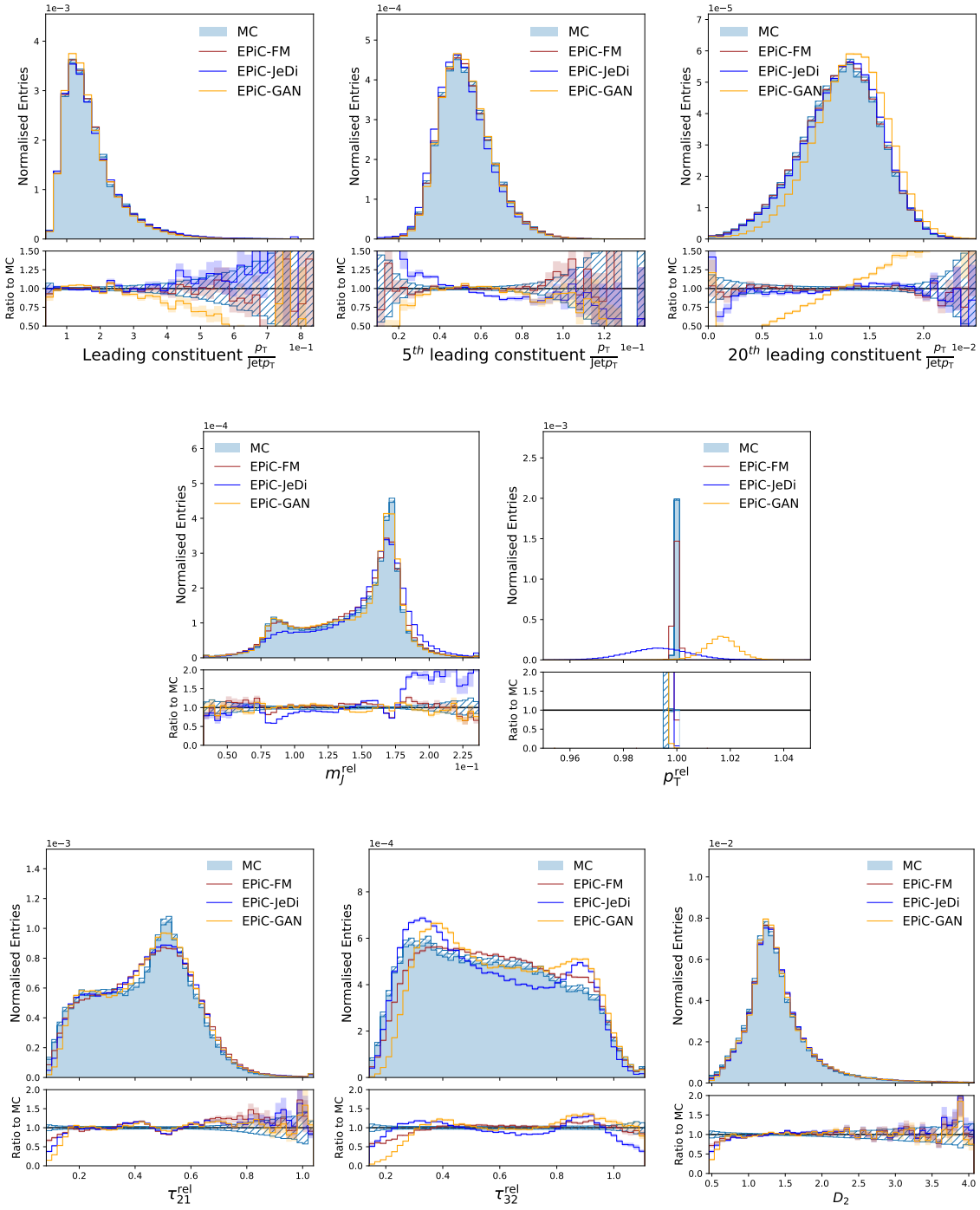


Figure 7.11: Unconditional top jet generation with 150 constituents: Various particle- and jet-level observables for the test set, EPiC-FM, EPiC-JeDi, and EPiC-GAN. **Top row:** relative p_T distribution of the leading (left), 5$^{\text{th}}$ leading (middle), and 20$^{\text{th}}$ leading (right) constituents. **Middle row:** relative jet mass (left) and jet p_T (right) distributions. **Bottom row:** relative jet τ_{21} (left), τ_{32} (middle), and D_2 (right) distributions. The error bands correspond to the statistical uncertainty of the bin count. Figures originally published in Reference [4].

the Wasserstein distance for generative modeling evaluation [62]. Yet, when comparing histograms in high-energy physics, one often cares about the matching of densities in the two distributions, i.e. an overdensity in a distribution might indicate a resonance signal among a large background. An example of the failure of a Wasserstein-based score to convey properly the performance of a distribution is given in Section 7.5.2.

To use the KLD consistently across various jet observables, we made the following binning choices: The binning range is set by the minimum and maximum values of the target (truth) distribution. A total of 100 bins are chosen and the bin edges are adjusted so that they result in equiprobable quantiles of the target distribution, i.e. equal bin content in each bin. No overflow bins are used, hence outliers in the model distributions are discarded — yet, since we compare equal-sized distributions, outliers lead to deviations in some model bins. This binning allows for equal statistical significance across the whole distribution and simplifies the KLD Equation 4.11 to:

$$D_{\mathrm{KL}}(P||Q) = -\frac{1}{N}\sum_{i=1}^{N}\left(\log(q_i) - \log\left(\frac{1}{N}\right)\right) \tag{7.4}$$

with Q as the model distribution and P as the uniform target ("truth") distribution with $p_{i\in[1,N]} = \frac{1}{N}$ in N bins. The $\mathrm{KL}^{\mathcal{O}}$ scores are hence equally sensitive to variations in all bins and in comparison to the Wasserstein distance not sensitive to the distance between these bins, i.e. mismodelings are equally weighted across the whole distribution and not depending on their relative distance.

We have a test statistic of about 27,000 jets and sampled 270,000 from each generative model. The $\mathrm{KL}^{\mathcal{O}}$ scores are calculated 40 times with 50,000 samples using bootstrapping. We report the mean and standard deviation of the $\mathrm{KL}^{\mathcal{O}}$ scores. As observables $\mathcal{O}$, we calculate the $\mathrm{KL}^{\mathcal{O}}$ scores with the relative jet mass, the jet constituent p_{T}/jet p_{T} distribution, the N-subjettiness ratios τ_{21} and τ_{32}, and the energy correlator D_2. In addition to the $\mathrm{KL}^{\mathcal{O}}$ scores, we calculate the Fréchet ParticleNet Distance (FPND) [62] (introduced in Section 7.3.1). For the JetNet150 dataset, we approximate the FPND by calculating it with the first 30 leading constituents.

JetNet30 Top Dataset

The resulting FPND and KLD scores for both the unconditional and the conditional versions of EPiC-FM and EPiC-JeDi as well as the scores for the EPiC-GAN (unconditional) and PC-JeDi (conditional) on the JetNet30 top dataset are shown in Table 7.5.

Comparing the unconditional models with each other, we observe that EPiC-FM reaches the lowest FPND score with the EPiC-GAN being only slightly worse. The scaling of this score is however difficult to interpret since it is unclear how the ParticleNet discriminator weights each feature. In all $\mathrm{KL}^{\mathcal{O}}$ scores, except the KL^{m}, the EPiC-FM model is performing the best and the EPiC-GAN slightly worse than EPiC-JeDi. In the KL^{m} score, the EPiC-GAN performs better than EPiC-FM, likely since the mass was used for the epoch choice. EPiC-JeDi performs significantly worse in the mass distribution, which is also clearly visible in Figure 7.10.

Comparing the conditional models with each other, we observe the best performance across all evaluation scores by the EPiC-FM model. Compared to PC-JeDi, the EPiC-JeDi

Table 7.5: Top jet generation with 30 constituents: summary of performance metrics for generated jets using the Fréchet ParticleNet Distance (FPND) and the Kullback-Leibler divergence for various particle- and jet observables. As comparisons to EPiC-FM and EPiC-JeDi (either with conditional or unconditional generation), the results of EPiC-GAN (unconditional) and PC-JeDi (conditional) are shown. Table adapted from Reference [4].

Generation	Model	FPND	$\mathrm{KL}^{m}(\times 10^{-3})$	$\mathrm{KL}^{p_T^{\mathrm{const}}}(\times 10^{-3})$	$\mathrm{KL}^{\tau_{21}}(\times 10^{-3})$	$\mathrm{KL}^{\tau_{32}}(\times 10^{-3})$	$\mathrm{KL}^{D_2}(\times 10^{-3})$
Unconditional	EPiC-GAN	0.34	$\mathbf{3.71 \pm 0.42}$	3.33 ± 0.03	8.28 ± 0.76	17.68 ± 0.91	13.18 ± 1.04
	EPiC-JeDi	1.63	18.42 ± 1.12	3.73 ± 0.08	8.00 ± 0.80	15.27 ± 1.35	12.33 ± 1.06
	EPiC-FM	**0.14**	5.80 ± 0.54	$\mathbf{2.03 \pm 0.01}$	$\mathbf{7.69 \pm 0.71}$	$\mathbf{9.24 \pm 1.00}$	$\mathbf{4.51 \pm 0.58}$
Conditional	PC-JeDi	0.40	8.56 ± 0.75	3.25 ± 0.09	12.82 ± 1.16	27.08 ± 1.40	11.91 ± 0.92
	EPiC-JeDi	0.42	5.26 ± 0.51	2.99 ± 0.05	7.81 ± 0.61	17.34 ± 1.08	6.58 ± 0.73
	EPiC-FM	**0.11**	$\mathbf{3.77 \pm 0.50}$	$\mathbf{2.03 \pm 0.02}$	$\mathbf{7.40 \pm 0.64}$	$\mathbf{8.09 \pm 0.93}$	$\mathbf{4.31 \pm 0.46}$

model yields lower $\mathrm{KL}^{\mathcal{O}}$ scores, however, in the FPND score, it is just slightly worse than PC-JeDi. Note that PC-JeDi uses the Euler-Maruyama (EM) solver as suggested in Reference [262]. Overall it appears that although PC-JeDi and EPiC-JeDi share the same diffusion parameterization, the simplified EPiC layer in combination with the midpoint solver is advantageous. Replacing the diffusion model with a flow matching CNF as in EPiC-FM leads to even better results.

Comparing the conditional and unconditional EPiC-FM with each other, we observe equal or slightly improved scores using the conditional model. In the mass score, we can see the largest improvement, which is to be expected since the model was specifically conditioned on the jet mass. One could argue that giving a model conditional information is "cheating" since one could in principle add a very large number of jet observables as conditioning to improve the model performance, i.e. all jet observables the models are evaluated on here. However, when the best-performing model is required, then such a conditional model might be exactly what should be used. Yet it is remarkable, that the EPiC-FM model even without conditioning performs very well and that the conditioning only leads to marginal improvements.

JetNet150 Top Dataset

The resulting FPND and KLD scores for both the unconditional and the conditional versions of EPiC-FM and EPiC-JeDi as well as the scores for the unconditional EPiC-GAN on the JetNet150 top dataset are shown in Table 7.6.

Comparing the unconditional models with each other, we observe a similar result as with the unconditional models on JetNet30. EPiC-FM is best in all scores except the mass score, where it is again outperformed by EPiC-GAN. EPiC-JeDi performs better than EPiC-GAN only in the $\mathrm{KL}^{\tau_{21}}$ and the $\mathrm{KL}^{\tau_{32}}$ scores, in the remaining scores EPiC-GAN achieves lower scores.

Note that the $\mathrm{KL}^{\tau_{32}}$ score is much lower for EPiC-FM ($\mathrm{KL}^{\tau_{32}} = 9.24 \pm 1.00$) than for EPiC-JeDi ($\mathrm{KL}^{\tau_{32}} = 15.27 \pm 1.35$). This aligns well with our observation of the τ_{32} distribution in Figure 7.10, where the shape of the EPiC-FM distribution resembles more closely the

Table 7.6: Top jet generation with 150 constituents: summary of performance metrics for generated jets using the Fréchet ParticleNet Distance (FPND) and the Kullback-Leibler divergence for various particle- and jet observables. As comparisons to EPiC-FM and EPiC-JeDi (either with conditional or unconditional generation) and the results of EPiC-GAN (unconditional) are shown. Table adapted from Reference [4].

Generation	Model	FPND	$\mathrm{KL}^{m}(\times 10^{-3})$	$\mathrm{KL}^{p_T^{\mathrm{const}}}(\times 10^{-3})$	$\mathrm{KL}^{\tau_{21}}(\times 10^{-3})$	$\mathrm{KL}^{\tau_{32}}(\times 10^{-3})$	$\mathrm{KL}^{D_2}(\times 10^{-3})$
Unconditional	EPiC-GAN	0.93	$\mathbf{6.50 \pm 0.63}$	2.22 ± 0.09	20.60 ± 1.55	69.64 ± 3.30	6.04 ± 0.64
	EPiC-JeDi	1.93	27.46 ± 1.24	6.39 ± 0.60	20.15 ± 1.25	36.50 ± 1.81	11.70 ± 0.98
	EPiC-FM	**0.18**	12.95 ± 0.90	$\mathbf{0.87 \pm 0.02}$	$\mathbf{10.59 \pm 0.88}$	$\mathbf{12.14 \pm 0.97}$	$\mathbf{4.39 \pm 0.55}$
Conditional	EPiC-JeDi	0.52	9.10 ± 0.79	6.42 ± 0.76	14.32 ± 1.08	19.92 ± 1.21	9.40 ± 0.88
	EPiC-FM	**0.12**	$\mathbf{4.30 \pm 0.53}$	$\mathbf{0.84 \pm 0.02}$	$\mathbf{9.43 \pm 0.61}$	$\mathbf{11.22 \pm 1.02}$	$\mathbf{4.28 \pm 0.56}$

MC distribution than EPiC-JeDi does. However, if we were to calculate the τ_{32} score based on the 1-Wasserstein distance, the resulting $W_1^{\tau_{32}}$ score would indicate that EPiC-JeDi ($W_1^{\tau_{32}} = 11.67 \pm 0.60$) performs better than EPiC-FM ($W_1^{\tau_{32}} = 19.85 \pm 1.29$). This is a result of the shape of the τ_{32} distribution where an underestimation of the center of the distribution is surrounded by an overestimation of the edges of the distribution, which cancels out in the Wasserstein score and leads to an overall better score for a seemingly worse distribution. This is an example of the failure mode of the Wasserstein scores compared to the KLD scores and shows that the KLD scores align more closely with a density-based interpretation of histograms.

Comparing the conditional models with each other, we observe the same result as with the unconditional models on JetNet30: EPiC-FM performs better than EPiC-JeDi across all scores. PC-JeDi was not available for this comparison since it would be too computationally costly to train on JetNet150. When comparing the conditional and the unconditional EPiC-FM models with each other, we can observe again that the conditioning improves the fidelity of the model leading to a better performance of the conditional EPiC-FM model across all scores.

7.5.3 Timing

An advantage of the EPiC-CNF models over PC-JeDi is the linear computational scaling with the point cloud size of the EPiC layers compared to the self-attention transformer layers employed by PC-JeDi, which scale quadratically. To benchmark the computational performance difference, we generated 270,000 jets with (fixed) constituent sizes between 10 and 150 with PC-JeDi and with EPiC-JeDi (which during inference has the identical computational requirements as EPiC-FM) on the same hardware with an NVIDIA® A100-40GB graphics card. Similar to the timing study for EPiC-GAN in Section 7.3.3, the batch size was adjusted for each fixed jet size. For this study, the models were randomly initialized and not trained. Note that i.e. the timing for a fixed 150 particles is not equal to the average timing of generating a JetNet150 jet, as most JetNet150 jets contain less than 100 constituents.

In Figure 7.12 we show the generation time per jet for each fixed generated particle

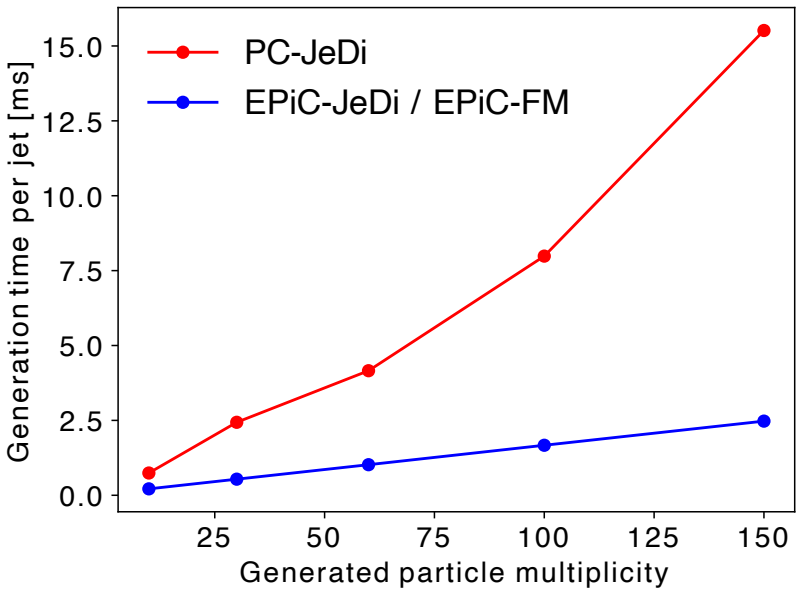


Figure 7.12: Timing performance of PC-JeDi compared to the EPiC-FM/EPiC-JeDi as a function of (fixed) generated particle multiplicity. As EPiC-JeDi and EPiC-FM use the same EPiC Network, the timing is representative of either approach. The batch size was optimized for optimal generation speed and generation was performed on a system with a single NVIDIA® A100-40GB GPU. The timing is calculated as an average of all test events generated. For reference, EPiC-GAN generates jets with 10 (150) particles in a little over 1 µs (10 µs) (see Figure 7.6). Figure originally published in Reference [4].

multiplicity of both PC-JeDi and EPiC-JeDi (EPiC-FM). The faster generation time of the EPiC-CNF models is apparent. Due to the $\mathcal{O}(N)$ scaling of the EPiC layers, the speed-up of the EPiC-CNF models over PC-JeDi is increasing with the point cloud size N. At 150 particles per jet, the EPiC-CNF models are 6.2× faster than PC-JeDi (2.5 ms vs. 15.5 ms). Overall, this much faster generation timing makes the EPiC-CNF models viable for modeling even larger point clouds, such as calorimeter showers as point clouds with $\mathcal{O}(1,000)$ points (see Chapter 6).

For comparison, the EPiC-GAN is about 210× faster than EPiC-CNF, generating 150 particles in just 12 μs. This speed-up is consistent as we generated jets using the midpoint ODE solver with 200 model evaluations. As the EPiC-GAN and the EPiC-CNF models share largely the same architecture, each EPiC-CNF model pass takes about as much time as a jet generation with EPiC-GAN. Yet, this speed-up comes at the cost of reduced generative fidelity with the EPiC-GAN. Distillation techniques for diffusion models, such as progressive distillation [218] and consistency distillation [219], could be applied to the EPiC-CNF models to further increase their computational efficiency. Hence, this work of speeding up the model architecture itself using the EPiC layers is orthogonal to work that has implemented distillation procedures for jet generative diffusion models using transformers [258, 263].

7.6 Summary

Many measurements among various sciences can be best represented as point clouds or sets of data. As point (particle) clouds are a natural representation for jets, the JetNet30 and JetNet150 datasets are specifically designed to develop point cloud generative models for particle physics applications. Apart from the potential to speed-up traditional Monte Carlo simulations, the generation of complex particle jets provides an interesting playground for developing point cloud generative models. Among the multiple jet types contained, we focus on the top jets with the most challenging mass distribution to model.

Motivated by the high computational requirements of graph-network and transformer-based point cloud generative models which limits their applicability to high cardinality point clouds (such as calorimeter showers), we introduce the equivariant point cloud (EPiC) layers. The EPiC layers are based on the Deep Sets principle, scale linearly with the size of the point cloud, and allow for variable-size point clouds. Using these layers allows us to develop the EPiC-GAN, a generative adversarial network, which can generate particle jets as point clouds.

We use the JetNet30 and the JetNet150 datasets to benchmark the performance of the EPiC-GAN against the (at the time) state-of-the-art MP-GAN. With both the qualitative evaluation of histograms of various particle- and jet-level observables and the quantitative evaluation of Wasserstein distance-based scores, we find comparable generative fidelity is reached by the MP-GAN and the EPiC-GAN.

While the two models reach a similar fidelity, the EPiC-GAN requires significantly fewer computational resources and can generate jets with fixed 150 particles 55× faster than the MP-GAN on the same GPU hardware. This linear scaling with the point cloud size allows for the application of the EPiC-GAN to the JetNet150 datasets as well as for possible future generation of even larger point clouds, such as calorimeter shower point clouds with $\mathcal{O}(1,000)$ cardinality. Additionally, the EPiC layers in the EPiC-GAN provide a global latent space, which can be correlated with physical observables allowing for an interpretation of the models' decision-making. This can help with establishing trust in the model predictions as well as with monitoring the physics encoding during training.

We further employ the EPiC layers to speed-up the PC-JeDi model, a recently introduced transformer-based diffusion model for jet generation. As a first step, we replace the transformer layers in PC-JeDi with EPiC layers resulting in the EPiC-JeDi model. As a second step, we replace the DDIM diffusion model training objective with optimal transport flow matching leading to the EPiC-FM model. Both models share the same architecture and the same sampling procedure, yet deviate in their training objective. Together we refer to them as EPiC-CNF, as they both resemble the same continuous normalizing flow (CNF) during sampling.

We compare the EPiC-CNF models with the EPiC-GAN and PC-JeDi on various differential distributions of particle and jet observables. To quantify their performance, we introduce the $\text{KL}^{\mathcal{O}}$ scores using the Kullback-Leibler divergence for the estimation of the similarity of model distributions to the Monte Carlo "truth" as an improvement over the previously used Wasserstein distance scores $W_1^{\mathcal{O}}$. The $\text{KL}^{\mathcal{O}}$ score is density-based and weights

dis-similarities between distributions equally across the whole parameter space instead of by their relative distance like the $W_1^{\mathcal{O}}$ scores. We see that especially for subtle differences like in the N-subjettiness ratios the $\mathrm{KL}^{\mathcal{O}}$ scores capture better a physicists' view of comparing histograms.

We find that across almost every metric on both datasets and for both conditional and unconditional models EPiC-FM performs best. EPiC-JeDi outperforms EPiC-GAN and PC-JeDi in many distributions, yet does not represent the test distributions as closely as EPiC-FM. This shows that on the one hand using the EPiC layers instead of transformer layers for a particle cloud generative diffusion model does not negatively impact the model performance and that the flow matching objective for training CNFs leads to superior results compared to the DDIM diffusion model parameterization.

In addition to the increased generative fidelity using the EPiC-FM model, it is also more computationally efficient than using transformer layers, as the EPiC layers scale linearly with the point cloud cardinality. Compared to PC-JeDi this leads to a 6.2× faster generation of jets with 150 particles with the same number of model evaluations. This improvement is complementary to research investigating distillation methods that aim to reduce the number of model evaluations [258, 263].

Overall, these results seem counter-intuitive since one would expect that message-passing and transformer layers have much more capacity for learning complex structures in the data compared to the simple EPiC layers. We speculate that since jet generation is a stochastic process with a relatively small number of important features, this simple architecture works still well, even though the inter-point correlations are only modeled by a small number of global feature vectors. Alternatively, the potentially smaller capacity of the EPiC layers improves the training as the overall gradient landscape might be simpler leading to an easier optimization. For future research into optimal layer structures for point cloud neural networks for HEP research it could be beneficial to disentangle the relative importance of point-, edge-, and global features.

However, none of the models reproduce exactly the test data in every single distribution, so at least some jets are still distinguishable from the MC "truth" distribution. While the research presented here provides an improvement in both fidelity and computational efficiency over previous approaches, further research should be performed to develop models with even higher fidelity and even greater speed-ups. However, for some applications, i.e. for large-scale parameter scans or proof-of-principle studies, it might be advantageous to use one of the presented generative models if the gains in computational efficiency outweigh the potentially lower fidelity.

In subsequent work [11], the EPiC-FM model was successfully applied to generate jets from the more complex JetClass [261] dataset and advanced the model by modeling more particle observables such as particle identification and track displacement information. Additionally, the EPiC-FM model was adapted [12] to explore anomaly detection with the R&D dataset of the LHC Olympics [286] using the full phase space in the CATHODE approach [287].

Chapter 8

Conclusion

The increasing luminosity and growing granularity of detector systems at current and future collider experiments motivate the development of more precise fast detector simulations. Generative machine learning models offer powerful and flexible modeling frameworks that are well suited for this task. In this thesis, multiple generative models for the simulation of calorimeter showers in highly-granular calorimeters as well as for the simulation of jets are presented and evaluated. This work advances the research on generative models for fast simulations in several ways:

In Chapter 5, the BIB-AE model for the generation of electromagnetic calorimeter showers is studied. The BIB-AE model unifies VAE- and GAN-based modeling approaches and generates calorimeter showers in the form of 3D images. It generates photon showers 10× faster than GEANT4 on the same CPU hardware. A comprehensive study of the latent space encoding is presented, which shows that only few variables are utilized to encode most of the information. These latent variables are highly correlated to observables such as the center of gravity in shower incident direction. This result can be used for targeted sampling for showers with specific properties, and it motivates an improvement of the latent space sampling via a kernel density estimate of the encoded latent space. By sampling from the KDE, correlations in the latent space are preserved, and the generative fidelity is increased.

Additionally, studies on the evaluation of generative models for calorimeter simulation are presented. The mean of 1D Wasserstein distances and a multi-dimenisonal MMD are found to be similarly sensitive to Gaussian perturbations in the shower dataset and very fast to compute. The AUC of a high-level classifier is a sensitive score on a wide range of Gaussian perturbation, especially compared to a low-level classifier that can already almost perfectly separate small perturbations and is therefore less suitable for comparing various generative models. The novel Fréchet Regression Distance is introduced, however it proves difficult to interpreted what shower properties contribute to a low FRD score.

In Chapter 6, three models for calorimeter shower generation as point clouds are introduced: CALOCLOUDS, CALOCLOUDS II, and CALOCLOUDS II (CM). Representing showers as point clouds has several advantages over voxelized representations: only hits with energy depositions are generated, the generation speed scales with the number of points, and the shower is largely geometry-independent. With CALOCLOUDS, we introduce a high-fidelity generative model for calorimeter showers as high-cardinality point clouds. The model is based on the

discrete-time DDPM diffusion model ansatz and achieves a modest speed up of 20% compared to GEANT4 on a CPU. With CALOCLOUDS II, we introduce an improved version based on continuous-time score matching, which leads to a significant speed of 6× compared to GEANT4 on the same CPU. Finally, we are able to distill the consistency model CALOCLOUDS II (CM), which achieves a speed up of 46× as well as the highest generative fidelity. The latter model constitutes the first consistency model for calorimeter shower generation and highlights the potential of consistency distillation for future generative models in high-energy physics.

In Chapter 7, we present two types of models for the generation of particle jets as point clouds. Both models are implemented with the novel equivariant point cloud (EPiC) layer, which is optimized for point cloud data, provides a global latent space for inter-point correlations and its computational requirement scales linear with the point cloud cardinality. Together with the EPiC layer, the EPiC-GAN model is introduced. It is evaluated on the JetNet benchmark dataset against the MP-GAN and we find that it provides a very similar generative fidelity, but generates jets with 150 particles 55× faster due to the superior scaling of the EPiC layer.

Additionally, EPiC-CNF is introduced, which is either trained with flow matching (EPiC-FM) or with score matching (EPiC-JeDi), and also utilizes the EPiC layer. Comparing the two training objectives, we find that EPiC-FM achieves the higher generative fidelity. It also outperforms the transformer-based diffusion model PC-JeDi in terms of both generative fidelity and generation speed. Both EPiC-CNF models achieve a higher fidelity than EPiC-GAN, but are significantly slower in generation speed. Overall, both the EPiC layer and the flow matching objective promise to be useful tools for future point cloud generative models in high-energy physics.

From here, several directions for future research can be identified:

Since the BIB-AE model was introduced, a novel but very similar class of generative models has emerged in the realm of image generation: latent diffusion models (LDMs) [288] contain the same building blocks as the BIB-AE – a VAE, an adversarial training loss, and a model for sampling the latent space. The difference is, that in the BIB-AE the latent space is sampled from a KDE (and in subsequent implementations from a normalizing flow) and in LDMs the latent space is sampled from a diffusion model. Over the recent years, LDMs have proven to be very powerful in generating high-quality images and it would be interesting if the already great generative fidelity of the BIB-AE could be further improved by incorporating the diffusion model sampling.

Although point cloud generative models have several advantages when it comes to the generation of calorimeter showers, they are not yet on par with image-based models in terms of generative fidelity. Future comparisons between the two model types, similar to Reference [249] and as pursued by the CaloChallenge [14], may help in identifying the best modeling practices and improve either model type. Future iterations of the CALOCLOUDS model could be based on the transformer architecture to allow for inter-point communication. This may be needed for an accurate modeling of hadron showers. However, it may come at the cost of generation speed and the trade-off between fidelity and speed should be carefully considered. Currently, efforts are made to apply the CALOCLOUDS model to hadronic showers,

to model showers in the CMS HGCAL, and to incorporate the model into the full GEANT4 simulation chain.

EPiC-FM has since been explored for its application in anomaly detection [12] and for the generation of jets in the JetClass dataset [11]. For future R&D of point cloud generative models in HEP, the JetClass dataset is a good candidate, as it is significantly larger than JetNet, includes more jet types, and more jet properties. With EPiC-FM, the flow matching objective is shown to be very effective for stable training of diffusion models / CNFs. It would be interesting to explore the distillation of EPiC-FM, e.g. into a consistency model, which could accelerate the generation process significantly. Recently, consistency trajectory models (CTMs) [289] were introduced as a unification of diffusion models and consistency models. Applying this modeling paradigm to fast simulation could be very promising. Overall, generative machine learning models for fast particle physics simulations will continue to be an active and interesting field of research in the coming years.

Appendix A

Supplementary Calorimeter Observables

Bin edges	0	1	2	3	4	5	6	7	8	9	10
Edges for $E_{\text{radial},i\in[1,10]}$ [mm]	0	6.6	9.8	13.0	17.0	23.4	33.6	40.1	48.5	68.8	300
Edges for $E_{\text{long},i\in[1,10]}$ [layer]	1	9	12	14	16	17	19	20	22	25	30

Table A.1: Bin ranges for calculating the radial and longitudinal energy observables $E_{\text{radial},i\in[1,10]}$ and $E_{\text{long},i\in[1,10]}$. They are determined on 40,000 events of the CaloClouds test dataset for ten quantiles each including on average the same number of cell hits. All bins are half-open, except the last bin. The bin edges are precisely calculated for the radial energy observables, but rounded to the nearest integer for the longitudinal ones. Table originally published in Reference [5].

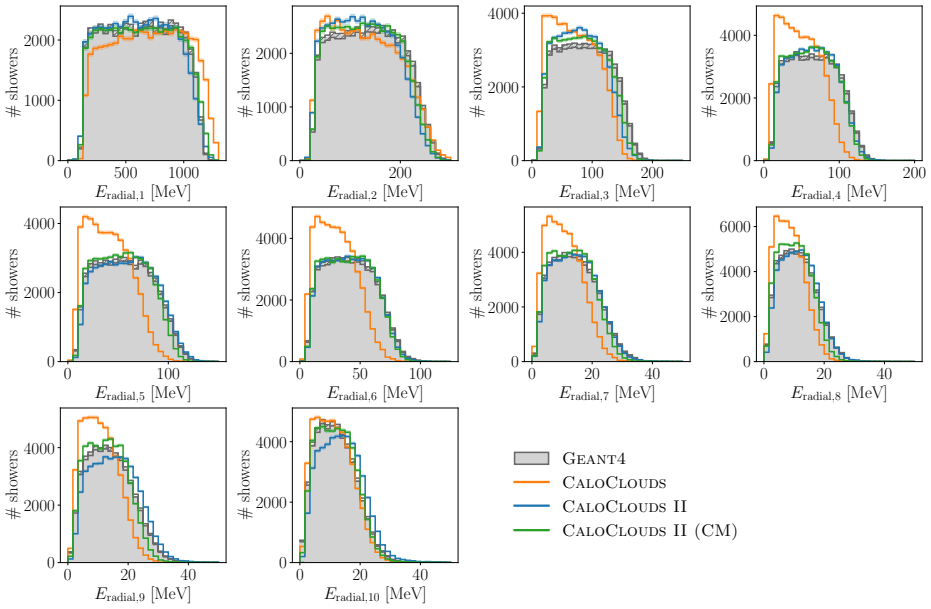


Figure A.1: Histograms of 10 radial energy observables comparing Geant4 to CaloClouds, CaloClouds II, and CaloClouds II (CM). A total of 50,000 showers are shown, sampled from a uniform distribution of incident particle energies between 10 and 90 GeV. The error band corresponds to the statistical uncertainty in each bin. Figure originally published in Reference [5].

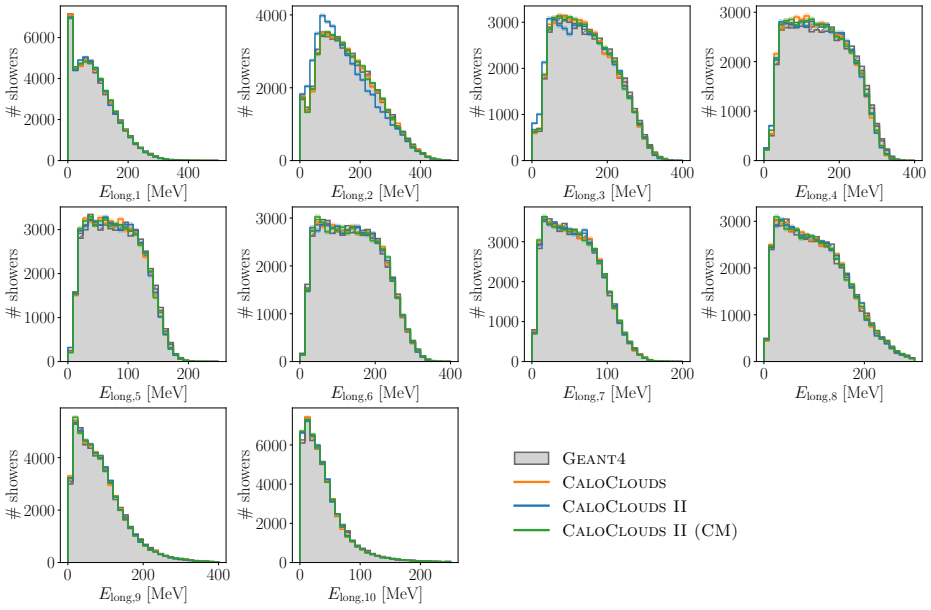


Figure A.2: Histograms of 10 longitudinal energy observables comparing Geant4 to CaloClouds, CaloClouds II, and CaloClouds II (CM). A total of 50,000 showers are shown, sampled from a uniform distribution of incident particle energies between 10 and 90 GeV. The error band corresponds to the statistical uncertainty in each bin. Figure originally published in Reference [5].

Acknowledgements

As a concluding remark, I would like to thank a people who have supported me throughout the years working on this thesis, although I might have forgotten some.

First and foremost, I would like to thank my doctoral advisor, Gregor Kasieczka, for his exemplary supervision of this thesis and for his innovative ideas and continuous support. His encouragements during challenging moments and his commitment to my reserach have been instrumental in shaping the direction and success of this thesis.

I would like to thank Frank Gaede for very interesting discussion in weekly meetings, for his enthusiam for novel methods, and for serving as the second examiner of this thesis.

I would like to thank Jesse Thaler for the time he took out of his busy schedule to discuss my research, for inviting me to the MIT, and for his continuous encouragement.

I would like to thank Daniela Pfannkuche, Erika Garutti, and Marcus Brüggen for being part of my examination committee.

I would like to thank Sascha Diefenbacher, Henry Day-Hall, Engin Eren, William Korcari, Anatolii Korol, Katja Krüger, Peter McKeown, Martina Mozzanica, and Lorenzo Valente for intersting weekly discussions on generative machine learning for particle physics which have shaped the direction of this thesis.

I would like to thank Sebastian Bieringer, Thorsten Buss, Sascha Diefenbacher, Cedric Ewen, William Korcari, and Martina Mozzanica for providing a great office experience and for the many interesting discussions.

I would like to thank the students I had the pleasure of supervising – Cedric Ewen, Malte Jacobsen, Jan Schreiber, and Nana Werther – for their hard work, quick learning, and insightful questions.

I would like to thank Joschka Birk, Thorsten Buss, Karim El Moabit, William Korcari, and Peter McKeown for proof-reading parts of this thesis and for providing valuable feedback.

I would like to thank all members of the GK group for their positive energy and for providing a supportive and fun working environment.

I would like to thank the Maxwell computing center at DESY for providing a great computing environment for training machine learning models on GPUs.

I would like to thank Martje for her unwavering support and patience while I finished this thesis and my whole family for their support throughout my years of study.

Finally, I would like to acknowledge and thank the Friedrich-Naumann-Foundation for Freedom for supporting me through a PhD scholarship funded by the Federal Minstry of Education and Research.

Thank you.

Bibliography

[1] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger 2021 **Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network** *EPJ Web Conf.* **251** 03003. e-Print: 2102.12491 doi: 10.1051/epjconf/202125103003

[2] E. Buhmann, G. Kasieczka and J. Thaler 2023 **EPiC-GAN: Equivariant point cloud generation for particle jets** *SciPost Physics* **15** doi: 10.21468/scipostphys.15.4.130

[3] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown 2023 **CaloClouds: fast geometry-independent highly-granular calorimeter simulation** *JINST* **18** P11025. e-Print: 2305.04847 doi: 10.1088/1748-0221/18/11/P11025

[4] E. Buhmann, C. Ewen, D. A. Faroughy, T. Golling, G. Kasieczka, M. Leigh, G. Quétant, J. A. Raine, D. Sengupta and D. Shih 2023 **EPiC-ly Fast Particle Cloud Generation with Flow-Matching and Diffusion**. e-Print: 2310.00049

[5] E. Buhmann, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown 2024 **CaloClouds II: ultra-fast geometry-independent highly-granular calorimeter simulation** *JINST* **19** P04020. e-Print: 2309.05704 doi: 10.1088/1748-0221/19/04/P04020

[6] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger 2021 **Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed** *Comput. Softw. Big Sci.* **5** 13. e-Print: 2005.05334 doi: 10.1007/s41781-021-00056-0

[7] E. Buhmann, S. Diefenbacher, E. Eren *et al.* 2021 **Fast and Accurate Electromagnetic and Hadronic Showers from Generative Models** *EPJ Web Conf.* **251** 03049 doi: 10.1051/epjconf/202125103049

[8] L. Benato, E. Buhmann, M. Erdmann, P. Fackeldey, J. Glombitza, N. Hartmann, G. Kasieczka, W. Korcari, T. Kuhr, J. Steinheimer, H. Stöcker, T. Plehn *et al.* 2022 **Shared Data and Algorithms for Deep Learning in Fundamental Physics** *Computing and Software for Big Science* **6** 9 doi: 10.1007/s41781-022-00082-6

[9] E. Buhmann, S. Diefenbacher, D. Hundhausen, G. Kasieczka, W. Korcari, E. Eren, F. Gaede, K. Krüger, P. McKeown and L. Rustige 2022 **Hadrons, better, faster, stronger** *Mach. Learn. Sci. Tech.* **3** 025014. e-Print: 2112.09709 doi: 10.1088/2632-2153/ac7848

[10] P. McKeown, S. Bieringer, E. Buhmann, S. D. Diefenbacher *et al.* 2022 **Generative Models for Fast Simulation of Electromagnetic and Hadronic Showers in Highly Granular Calorimeters** *PoS* **ICHEP2022** 236 doi: 10.22323/1.414.0236

[11] J. Birk, E. Buhmann, C. Ewen, G. Kasieczka and D. Shih 2023 **Flow Matching Beyond Kinematics: Generating Jets with Particle-ID and Trajectory Displacement Information**. e-Print: 2312.00123 URL: https://arxiv.org/abs/2312.00123

[12] E. Buhmann, C. Ewen, G. Kasieczka, V. Mikuni, B. Nachman and D. Shih 2024 **Full phase space resonant anomaly detection** *Phys. Rev. D* **109** 055015. e-Print: 2310.06897 doi: 10.1103/PhysRevD.109.055015

[13] P. McKeown, E. Buhmann *et al.* 2024 **Fast Simulation of Highly Granular Calorimeters with Generative Models: Towards a First Physics Application** *PoS* **EPS-HEP2023** 568 doi: 10.22323/1.449.0568

[14] C. Krause, M. F. Giannelli, G. Kasieczka *et al.* 2024 **CaloChallenge 2022: A Community Challenge for Fast Calorimeter Simulation** *in preparation*

[15] M. Jacobsen 2021 **The Blessing of Dimensionality: Event Manifold Dimensionality Estimation for Event Clustering** Master's thesis Universität Hamburg

[16] N. M. Werther 2022 **Frechet Distance Evaluation of Generative Models for Calorimeter Shower Simulation** Master's thesis Universität Hamburg

[17] J. Schreiber 2023 **Evaluation of Evaluation Metrics of Generative Models in High Energy Physics** Master's thesis Universität Hamburg

[18] C. Ewen 2023 **Frechet Distance Evaluation of Generative Models for Calorimeter Shower Simulation** Master's thesis Universität Hamburg

[19] The ATLAS collaboration 2012 **A Particle Consistent with the Higgs Boson Observed with the ATLAS Detector at the Large Hadron Collider** *Science* **338** 1576–1582 doi: 10.1126/science.1232005

[20] The CMS Collaboration 2012 **Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC** *Physics Letters B* **716** 30–61 doi: 10.1016/j.physletb.2012.08.021

[21] I. Zurbano Fernandez *et al.* edited by I. Béjar Alonso, O. Brüning, P. Fessia, L. Rossi, L. Tavian and M. Zerlauth 2020 **High-Luminosity Large Hadron Collider (HL-LHC): Technical design report 10/2020** doi: 10.23731/CYRM-2020-0010

[22] 2017 **The Phase-2 Upgrade of the CMS Endcap Calorimeter** doi: 10.17181/CERN.IV8M.1JY2

[23] 2020 **2020 Update of the European Strategy for Particle Physics (Brochure)** Tech. rep. Geneva doi: 10.17181/CERN.JSC6.W89E URL: https://cds.cern.ch/record/2721370

[24] T. Behnke, J. E. Brau, B. Foster, J. Fuster, M. Harrison, J. M. Paterson, M. Peskin, M. Stanitzki, N. Walker and H. Yamamoto 2013 **The International Linear Collider Technical Design Report - Volume 1: Executive Summary** e-Print: 1306.6327 doi: 10.48550/arXiv.1306.6327

[25] M. Thomson 2009 **Particle flow calorimetry and the PandoraPFA algorithm** *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **611** 25–40 doi: https://doi.org/10.1016/j.nima.2009.09.009

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

[27] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. R. de Austri and R. Verheyen 2021 **Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer**. e-Print: 1901.00875

[28] B. Hashemi *et al.* 2019 **LHC analysis-specific datasets with Generative Adversarial Networks**. e-Print: 1901.05282

[29] R. Di Sipio *et al.* 2019 **DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC** *JHEP* **08** 110 doi: 10.1007/JHEP08(2019)110

[30] A. Butter, T. Plehn and R. Winterhalder 2019 **How to GAN LHC Events** *SciPost Phys.* **7** 075 doi: 10.21468/SciPostPhys.7.6.075

[31] J. Arjona Martinez *et al.* 2020 **Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description** *J. Phys. Conf. Ser.* **1525** 012081 doi: 10.1088/1742-6596/1525/1/012081

[32] C. Gao *et al.* 2020 **Event Generation with Normalizing Flows** *Phys. Rev. D* **101** 076002 doi: 10.1103/PhysRevD.101.076002

[33] Y. Alanazi *et al.* 2020 **Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)** doi: 10.24963/ijcai.2021/293

[34] M. Bellagente *et al.* 2020 **Invertible Networks or Partons to Detector and Back Again** *SciPost Phys.* **9** 074 doi: 10.21468/SciPostPhys.9.5.074

[35] L. Velasco *et al.* 2020 **cFAT-GAN: Conditional Simulation of Electron-Proton Scattering Events with Variate Beam Energies by a Feature Augmented and Transformed Generative Adversarial Network** *19th IEEE International Conference on Machine Learning and Applications* pp 372–375 doi: 10.1109/icmla51294.2020.00066

[36] A. Butter and T. Plehn 2020 **Generative Networks for LHC events** e-Print: 2008.08558 doi: 10.48550/arxiv.2008.08558

[37] J. N. Howard *et al.* 2022 **Learning to simulate high energy particle collisions from unlabeled data** *Sci. Rep.* **12** 7567 doi: 10.1038/s41598-022-10966-7

[38] G. Quétant *et al.* 2021 **Turbo-Sim: a generalised generative model with a physical latent space**. e-Print: 2112.10629

[39] M. Paganini, L. de Oliveira and B. Nachman 2018 **Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters** *Phys. Rev. Lett.* **120** 042003. e-Print: 1705.02355 doi: 10.1103/PhysRevLett.120.042003

[40] M. Paganini, L. de Oliveira and B. Nachman 2018 **CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks** *Phys. Rev. D* **97** 014021. e-Print: 1712.10321 doi: 10.1103/PhysRevD.97.014021

[41] L. de Oliveira, M. Paganini and B. Nachman 2018 **Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters** *J. Phys. Conf. Ser.* **1085** 042017. e-Print: 1711.08813 doi: 10.1088/1742-6596/1085/4/042017

[42] M. Erdmann, L. Geiger, J. Glombitza and D. Schmidt 2018 **Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks** *Comput. Softw. Big Sci.* **2** 4. e-Print: 1802.03325 doi: 10.1007/s41781-018-0008-x

[43] M. Erdmann, J. Glombitza and T. Quast 2019 **Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network** *Comput. Softw. Big Sci.* **3** 4. e-Print: 1807.01954 doi: 10.1007/s41781-018-0019-7

[44] S. Vallecorsa 2018 **Generative models for fast simulation** *J. Phys. Conf. Ser.* **1085** 022005 doi: 10.1088/1742-6596/1085/2/022005

[45] F. Carminati, A. Gheata, G. Khattak, P. Mendez Lorenzo, S. Sharan and S. Vallecorsa 2018 **Three dimensional Generative Adversarial Networks for fast simulation** *J. Phys. Conf. Ser.* **1085** 032016 doi: 10.1088/1742-6596/1085/3/032016

[46] P. Musella and F. Pandolfi 2018 **Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks** *Comput. Softw. Big Sci.* **2** 8. e-Print: 1805.00850 doi: 10.1007/s41781-018-0015-y

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

[48] H. Hashemi, N. Hartmann, S. Sharifzadeh, J. Kahn and T. Kuhr 2023 **Ultra-High-Resolution Detector Simulation with Intra-Event Aware GAN and Self-Supervised Relational Reasoning** e-Print: 2303.08046

[49] A. Collaboration 2022 **Deep generative models for fast photon shower simulation in ATLAS**. e-Print: 2210.06204

[50] J. C. Cresswell, B. L. Ross, G. Loaiza-Ganem, H. Reyes-Gonzalez, M. Letizia and A. L. Caterini 2022 **CaloMan: Fast generation of calorimeter showers with density estimation on learned manifolds**. e-Print: 2211.15380

[51] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, K. Krüger, P. McKeown and L. Rustige 2023 **New Angles on Fast Calorimeter Shower Simulation** e-Print: 2303.18150

[52] C. Chen, O. Cerri, T. Q. Nguyen, J. R. Vlimant and M. Pierini 2021 **Analysis-Specific Fast Simulation at the LHC with Deep Learning** *Computing and Software for Big Science* **5** 15 doi: 10.1007/s41781-021-00060-4

[53] C. Krause and D. Shih 2021 **CaloFlow II: Even Faster and Still Accurate Generation of Calorimeter Showers with Normalizing Flows** e-Print: 2110.11377 doi: 10.48550/arXiv.2110.11377

[54] C. Krause and D. Shih 2021 **CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows** e-Print: 2106.05285 doi: 10.48550/arXiv.2106.05285

[55] C. Krause, I. Pang and D. Shih 2024 **CaloFlow for CaloChallenge dataset 1** *SciPost Phys.* **16** 126. e-Print: 2210.14245 doi: 10.21468/SciPostPhys.16.5.126

[56] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh and D. Shih 2023 **L2LFlows: Generating High-Fidelity 3D Calorimeter Images** e-Print: 2302.11594

[57] A. Xu, S. Han, X. Ju and H. Wang 2023 **Generative Machine Learning for Detector Response Modeling with a Conditional Normalizing Flow**. e-Print: 2303.10148

[58] M. R. Buckley, C. Krause, I. Pang and D. Shih 2024 **Inductive simulation of calorimeter showers with normalizing flows** *Phys. Rev. D* **109** 033006. e-Print: 2305.11934 doi: 10.1103/PhysRevD.109.033006

[59] V. Mikuni and B. Nachman 2022 **Score-based generative models for calorimeter shower simulation** *Phys. Rev. D* **106** 092009. e-Print: 2206.11898 doi: 10.1103/PhysRevD.106.092009

[60] V. Mikuni and B. Nachman 2024 **CaloScore v2: single-shot calorimeter shower simulation with diffusion models** *JINST* **19** P02001. e-Print: 2308.03847 doi: 10.1088/1748-0221/19/02/P02001

[61] O. Amram and K. Pedro 2023 **CaloDiffusion with GLaM for High Fidelity Calorimeter Simulation**. e-Print: 2308.03876

[62] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant and D. Gunopulos 2021 **Particle Cloud Generation with Message Passing Generative Adversarial Networks** *35th Conference on Neural Information Processing Systems*. e-Print: 2106.11535

[63] J. J. Thomson 1897 **Cathode rays** *Phil. Mag. Ser. 5* **44** 293–316 doi: 10.1080/14786449708621070

[64] E. Rutherford 1911 **The scattering of alpha and beta particles by matter and the structure of the atom** *Phil. Mag. Ser. 6* **21** 669–688 doi: 10.1080/14786440508637080

[65] N. Bohr 1913 **On the Constitution of Atoms and Molecules** *Phil. Mag. Ser. 6* **26** 1–24 doi: 10.1080/14786441308634955

[66] J. Chadwick 1932 **Possible Existence of a Neutron** *Nature* **129** 312 doi: 10.1038/129312a0

[67] D. J. Griffiths 2008 **Introduction to Elementary Particles; 2nd rev. version** Physics textbook (New York, NY: Wiley)

[68] M. Thomson 2013 **Modern particle physics** (New York: Cambridge University Press) ISBN 978-1-107-03426-6 doi: 10.1017/CBO9781139525367

[69] W. N. Cottingham and D. A. Greenwood 2007 **An Introduction to the Standard Model of Particle Physics** (Cambridge University Press) ISBN 0521852498

[70] G. Kane 2017 **Modern Elementary Particle Physics: Explaining and Extending the Standard Model** 2nd ed (Cambridge University Press)

[71] R. Feynman and A. Zee 1985 **QED: The Strange Theory of Light and Matter** *Princeton University Press*

[72] H. Fritzsch, M. Gell-Mann and H. Leutwyler 1973 **Advantages of the color octet gluon picture** *Physics Letters B* **47** 365–368 doi: 10.1016/0370-2693(73)90625-4

[73] S. L. Glashow 1959 **The renormalizability of vector meson interactions** *Nuclear Physics* **10** 107–117 doi: 10.1016/0029-5582(59)90196-8

[74] S. Weinberg 1967 **A Model of Leptons** *Physical Review Letters* **19** 1264–1266 doi: 10.1103/physrevlett.19.1264

[75] C. N. Yang and R. L. Mills 1954 **Conservation of Isotopic Spin and Isotopic Gauge Invariance** *Phys. Rev.* **96**(1) 191–195 doi: 10.1103/PhysRev.96.191

[76] R. L. Workman and Others (Particle Data Group) 2022 **Review of Particle Physics** *PTEP* **2022** 083C01 doi: 10.1093/ptep/ptac097

[77] E. Brianne 2018 **Time Development of Hadronic Showers in a Highly Granular Analog Hadron Calorimeter** Ph.D. thesis University of Hamburg

[78] T. G. C. on Weights and M. (CGPM)" **On the revision of the International System of Units (SI)**

[79] H. Yukawa 1955 **On the Interaction of Elementary Particles. I** *Progress of Theoretical Physics Supplement* **1** 1–10 doi: 10.1143/PTPS.1.1

[80] D. J. Gross and F. Wilczek 1973 **Ultraviolet Behavior of Non-Abelian Gauge Theories** *Phys. Rev. Lett.* **30** 1343–1346 doi: 10.1103/PhysRevLett.30.1343

[81] H. D. Politzer 1973 **Reliable Perturbative Results for Strong Interactions?** *Phys. Rev. Lett.* **30** 1346–1349 doi: 10.1103/PhysRevLett.30.1346

[82] P. W. Higgs 1964 **Broken Symmetries and the Masses of Gauge Bosons** *Physical Review Letters* **13** 508–509

[83] F. Englert and R. Brout 1964 **Broken Symmetry and the Mass of Gauge Vector Mesons** *Physical Review Letters* **13** 321–323

[84] G. S. Guralnik, C. R. Hagen and T. W. B. Kibble 1964 **Global Conservation Laws and Massless Particles** *Phys. Rev. Lett.* **13**(20) 585–587 doi: 10.1103/PhysRevLett.13.585

[85] The UA1 Collaboration 1983 **Experimental observation of lepton pairs of invariant mass around 95 GeV/c2 at the CERN SPS collider** *Physics Letters B* **126** 398–410 doi: https://doi.org/10.1016/0370-2693(83)90188-0

[86] The UA2 Collaboration 1983 **Evidence for Z0→e+e- at the CERN pp collider** *Physics Letters B* **129** 130–140 doi: https://doi.org/10.1016/0370-2693(83)90744-X

[87] The UA1 Collaboration 1983 **Experimental observation of isolated large transverse energy electrons with associated missing energy at s=540 GeV** *Physics Letters B* **122** 103–116 doi: https://doi.org/10.1016/0370-2693(83)91177-2

[88] The UA2 Collaboration 1983 **Observation of single isolated electrons of high transverse momentum in events with missing transverse energy at the CERN pp collider** *Physics Letters B* **122** 476–485 doi: https://doi.org/10.1016/0370-2693(83)91605-2

[89] The TASSO Collaboration (TASSO) 1979 **Evidence for Planar Events in e+ e- Annihilation at High-Energies** *Phys. Lett. B* **86** 243–249 doi: 10.1016/0370-2693(79)90830-X

[90] A. Einstein 1916 **Die Grundlage der allgemeinen Relativitätstheorie** *Annalen der Physik* **354** 769–822 doi: 10.1002/andp.19163540702

[91] The Super-Kamiokande Collaboration (Super-Kamiokande Collaboration) 1998 **Evidence for Oscillation of Atmospheric Neutrinos** *Phys. Rev. Lett.* **81**(8) 1562–1567 doi: 10.1103/PhysRevLett.81.1562

[92] The SNO Collaboration (SNO Collaboration) 2002 **Direct Evidence for Neutrino Flavor Transformation from Neutral-Current Interactions in the Sudbury Neutrino Observatory** *Phys. Rev. Lett.* **89**(1) 011301 doi: 10.1103/PhysRevLett.89.011301

[93] The KATRIN Collaboration (Katrin) 2024 **Direct neutrino-mass measurement based on 259 days of KATRIN data** e-Print: 2406.13516

[94] G. Hooft 1980 **Naturalness, Chiral Symmetry, and Spontaneous Chiral Symmetry Breaking** (Boston, MA: Springer US) pp 135–157 ISBN 978-1-4684-7571-5 doi: 10.1007/978-1-4684-7571-5_9

[95] S. P. MARTIN 1998 **A SUPERSYMMETRY PRIMER** (WORLD SCIENTIFIC) pp 1–98 doi: 10.1142/9789812839657_0001

[96] H. Wiedemann 2015 **Particle Accelerator Physics** Graduate Texts in Physics (Cham: Springer International Publishing) ISBN 978-3-319-18316-9 978-3-319-18317-6 doi: 10.1007/978-3-319-18317-6 URL: http://link.springer.com/10.1007/978-3-319-18317-6

[97] L. Evans and P. Bryant 2008 **LHC Machine** *Journal of Instrumentation* **3** S08001 doi: 10.1088/1748-0221/3/08/S08001

[98] The CMS Collaboration (CMS) 2008 **The CMS Experiment at the CERN LHC** *JINST* **3** S08004 doi: 10.1088/1748-0221/3/08/S08004

[99] The Tracker Group of the CMS collaboration 2021 **The CMS Phase-1 pixel detector upgrade** *Journal of Instrumentation* **16** P02027 doi: 10.1088/1748-0221/16/02/P02027

[100] The CMS Collaboration (CMS) 2017 **The CMS trigger system** *JINST* **12** P01020. e-Print: 1609.02366 doi: 10.1088/1748-0221/12/01/P01020

[101] The CMS Collaboration edited by D. Contardo, M. Klute, J. Mans, L. Silvestris and J. N. Butler 2015 **Technical Proposal for the Phase-II Upgrade of the CMS Detector** doi: 10.17181/CERN.VU8I.D59J

[102] the ILC International Development Team and the ILC community 2023 **The International Linear Collider: Report to Snowmass 2021**. e-Print: 2203.07622 URL: https://arxiv.org/abs/2203.07622

[103] L. Linssen, A. Miyamoto, M. Stanitzki and H. Weerts 2012 **Physics and Detectors at CLIC: CLIC Conceptual Design Report**. e-Print: 1202.5940 URL: https://arxiv.org/abs/1202.5940

[104] D. Arominski, J.-J. Blaising, E. Brondolin, D. Dannheim, K. Elsener, F. Gaede, I. García-García, S. Green, D. Hynds, E. Leogrande, L. Linssen, J. Marshall *et al.* 2018 **A detector for CLIC: main parameters and performance**. e-Print: 1812.07337 URL: https://arxiv.org/abs/1812.07337

[105] The FCC Collaboration 2019 **FCC Physics Opportunities** *The European Physical Journal C* **79** 474 doi: 10.1140/epjc/s10052-019-6904-3

[106] The FCC Collaboration 2019 **FCC-ee: The Lepton Collider** *The European Physical Journal Special Topics* **228** 261–623 doi: 10.1140/epjst/e2019-900045-4

[107] The CEPC Study Group 2018 **CEPC Conceptual Design Report: Volume 2 - Physics & Detector**. e-Print: 1811.10545 URL: https://arxiv.org/abs/1811.10545

[108] The CEPC Study Group 2024 **CEPC Technical Design Report – Accelerator (v2)**. e-Print: 2312.14363 URL: https://arxiv.org/abs/2312.14363

[109] H. Al Ali, N. Arkani-Hamed, I. Banta, S. Benevedes, D. Buttazzo, T. Cai, J. Cheng, T. Cohen, N. Craig, M. Ekhterachian, J. Fan, M. Forslund *et al.* 2022 **The muon Smasher's guide** *Reports on Progress in Physics* **85** 084201 doi: 10.1088/1361-6633/ac6678

[110] K. L. Bane, T. L. Barklow, M. Breidenbach, C. P. Burkhart, E. A. Fauve, A. R. Gold, V. Heloin, Z. Li, E. A. Nanni, M. Nasr, M. Oriunno, J. M. Paterson *et al.* 2019 **An Advanced NCRF Linac Concept for a High Energy e^+e^- Linear Collider**. e-Print: 1807.10195 URL: https://arxiv.org/abs/1807.10195

[111] R. D. Ruth, A. W. Chao, P. L. Morton and P. B. Wilson 1985 **A PLASMA WAKE FIELD ACCELERATOR** *Part. Accel.* **17** 171

[112] B. Foster, R. D'Arcy and C. A. Lindstrøm 2023 **A hybrid, asymmetric, linear Higgs factory based on plasma-wakefield and radio-frequency acceleration** *New Journal of Physics* **25** 093037 doi: 10.1088/1367-2630/acf395

[113] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli and M. Zaro 2014 **The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations** *Journal of High Energy Physics* **2014** doi: 10.1007/jhep07(2014)079

[114] W. Kilian, T. Ohl and J. Reuter 2011 **WHIZARD—simulating multi-particle processes at LHC and ILC** *The European Physical Journal C* **71** doi: 10.1140/epjc/s10052-011-1742-y

[115] T. Sjöstrand, S. Mrenna and P. Skands 2008 **A brief introduction to PYTHIA 8.1** *Comput. Phys. Commun.* **178** 852–867 doi: 10.1016/j.cpc.2008.01.036

[116] J. Bellm *et al.* 2016 **Herwig 7.0/Herwig++ 3.0 release note** *Eur. Phys. J. C* **76** 196 doi: 10.1140/epjc/s10052-016-4018-8

[117] M. Frank, F. Gaede, C. Grefe and P. Mato 2014 **DD4hep: A Detector Description Toolkit for High Energy Physics Experiments** *Journal of Physics: Conference Series* **513** 022010 doi: 10.1088/1742-6596/513/2/022010

[118] S. Agostinelli *et al.* (GEANT4) 2003 **GEANT4–a simulation toolkit** *Nucl. Instrum. Meth. A* **506** 250 doi: 10.1016/S0168-9002(03)01368-8

[119] **HEP-SPEC06 Benchmark** URL: https://w3.hepix.org/benchmarking.html

[120] 2022 **ATLAS Software and Computing HL-LHC Roadmap** Tech. rep. CERN Geneva URL: https://cds.cern.ch/record/2802918

[121] K. Rabbertz 2018 **Jet Phyiscs at the LHC** (Sprinter Cham)

[122] C. Bierlich, S. Chakraborty, N. Desai, L. Gellersen, I. Helenius, P. Ilten, L. Lönnblad, S. Mrenna, S. Prestel, C. T. Preuss, T. Sjöstrand, P. Skands *et al.* 2022 **A comprehensive guide to the physics and usage of PYTHIA 8.3**. e-Print: 2203.11601 URL: https://arxiv.org/abs/2203.11601

[123] B. Andersson, G. Gustafson, G. Ingelman and T. Sjostrand 1983 **Parton Fragmentation and String Dynamics** *Phys. Rept.* **97** 31–145 doi: 10.1016/0370-1573(83)90080-7

[124] S. D. Ellis and D. E. Soper 1993 **Successive combination jet algorithm for hadron collisions** *Phys. Rev. D* **48** 3160–3166. e-Print: hep-ph/9305266 doi: 10.1103/PhysRevD.48.3160

[125] M. Wobisch and T. Wengler 1998 **Hadronization corrections to jet cross-sections in deep inelastic scattering** *Workshop on Monte Carlo Generators for HERA Physics (Plenary Starting Meeting)* pp 270–279. e-Print: hep-ph/9907280

[126] M. Cacciari, G. P. Salam and G. Soyez 2008 **The anti-k_t jet clustering algorithm** *JHEP* **04** 063. e-Print: 0802.1189 doi: 10.1088/1126-6708/2008/04/063

[127] J. Aparisi, I. García, M. Perelló, P. Roloff, R. Simoniello and M. Vos 2017 **Jet reconstruction algorithms in e^+e^- collisions** *Parton radiation and fragmentation from LHC to FCC-ee* pp 128–133

[128] M. Boronat, J. Fuster, I. García, E. Ros and M. Vos 2015 **A robust jet reconstruction algorithm for high-energy lepton colliders** *Physics Letters B* **750** 95–99 doi: 10.1016/j.physletb.2015.08.055

[129] S. Marzani, G. Soyez and M. Spannowsky 2019 **Looking Inside Jets: An Introduction to Jet Substructure and Boosted-object Phenomenology** (Springer International Publishing) ISBN 9783030157098 doi: 10.1007/978-3-030-15709-8 URL: http://dx.doi.org/10.1007/978-3-030-15709-8

[130] J. Thaler and K. Van Tilburg 2011 **Identifying boosted objects with N-subjettiness** *Journal of High Energy Physics* **2011** doi: 10.1007/jhep03(2011)015

[131] P. T. Komiske, E. M. Metodiev and J. Thaler 2018 **Energy flow polynomials: a complete linear basis for jet substructure** *Journal of High Energy Physics* **2018** doi: 10.1007/jhep04(2018)013

[132] A. J. Larkoski, G. P. Salam and J. Thaler 2013 **Energy correlation functions for jet substructure** *Journal of High Energy Physics* **2013** doi: 10.1007/jhep06(2013)108

[133] R. Wigmans 2008 **Calorimetry** *Scientifica Acta 2, No. 1, 18 - 55*

[134] M. Livan and R. Wigmans 2019 **Calorimetry for Collider Physics, an Introduction** (Springer Cham)

[135] R. Wigmans 2017 **Calorimetry: Energy Measurement in Particle Physics** International series of monographs on physics (Oxford University Press) ISBN 9780198786351 URL: https://books.google.de/books?id=BJg4DwAAQBAJ

[136] B. Rossi 1952 **High-energy Particles** Prentice-Hall physics series (Prentice-Hall) URL: https://books.google.de/books?id=wLQmAAAAMAAJ

[137] Y. Israeli 2018 **Energy Reconstruction in Highly Granular Calorimeters for Future Electron-Positron Colliders** Ph.D. thesis Technische Universität München

[138] The Super-Kamiokande Collaboration (Super-Kamiokande) edited by V. A. Ilyin, V. V. Korenkov and D. Perret-Gallix 2003 **The Super-Kamiokande detector** *Nucl. Instrum. Meth. A* **501** 418–462 doi: 10.1016/S0168-9002(03)00425-X

[139] The IceCube Collaboration 2017 **The IceCube Neutrino Observatory: instrumentation and online systems** *Journal of Instrumentation* **12** P03012–P03012 doi: 10.1088/1748-0221/12/03/p03012

[140] A. Bernstein *et al.* 1993 **Beam tests of the ZEUS barrel calorimeter** *Nuclear Instruments and Methods m Physics Research A 336 (1993) 23-52*

[141] R. Wigmans (DREAM) edited by G. Chiarelli, F. Cervelli, F. Forti and A. Scribano 2010 **The DREAM project: Towards the ultimate in calorimetry** *Nucl. Instrum. Meth. A* **617** 129–133 doi: 10.1016/j.nima.2009.09.118

[142] The H1 Collaboration (H1) 1990 **Software compensation for single particles and jets in the H1 calorimeter** *ECFA Large Hadron Collider (LHC) Workshop: Physics and Instrumentation*

[143] M. Thomson 2013 **Particle Flow Calorimetry** Invited Talk, "Physics at the Terascale", Mainz, Germany

[144] The CMS Collaboration 2017 **Particle-flow reconstruction and global event description with the CMS detector** *Journal of Instrumentation* **12** P10003–P10003 doi: 10.1088/1748-0221/12/10/p10003

[145] The Geant4 Collaboration 2023 **Geant4 — Physics Reference Manual, Release 11.2** https://geant4-userdoc.web.cern.ch/UsersGuides/PhysicsReferenceManual/fo/PhysicsReferenceManual.pdf

[146] G. Folger and J. P. Wellisch 2003 **String Parton Models in Geant4**. e-Print: nucl-th/0306007 URL: https://arxiv.org/abs/nucl-th/0306007

[147] D. Wright and M. Kelsey 2015 **The Geant4 Bertini Cascade** *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **804** 175–188 doi: https://doi.org/10.1016/j.nima.2015.09.058

[148] J. de Favereau *et al.* 2014 **DELPHES 3, A modular framework for fast simulation of a generic collider experiment** *JHEP* **02** 057 doi: 10.1007/jhep02(2014)057

[149] G. Grindhammer, M. Rudowicz and S. Peters 1990 **The fast simulation of electromagnetic and hadronic showers** *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **290** 469–488 doi: https://doi.org/10.1016/0168-9002(90)90566-O

[150] G. Grindhammer and S. Peters 1993 **The Parameterized simulation of electromagnetic showers in homogeneous and sampling calorimeters** *International Conference on Monte Carlo Simulation in High-Energy and Nuclear Physics - MC 93*. e-Print: hep-ex/0001020

[151] F. Wolf 2022 **Performance comparisons of generative ML and parameterized shower simulations for highly granular calorimeters** Master's thesis Universität Hamburg

[152] E. Barberio *et al.* edited by M. Fraternali, G. Gaudio and M. Livan 2009 **Fast simulation of electromagnetic showers in the ATLAS calorimeter: Frozen showers** *J. Phys. Conf. Ser.* **160** 012082 doi: 10.1088/1742-6596/160/1/012082

[153] S. D. Diefenbacher 2022 **Topics in Generative Modeling of Particle Physics Data** Ph.D. thesis Universität Hamburg

[154] ATLAS Collaboration 2022 **AtlFast3: The Next Generation of Fast Simulation in ATLAS** *Comput. Softw. Big Sci.* **6** doi: https://doi.org/10.1007/s41781-021-00079-7

[155] P. McKeown 2024 **Development and Performance of a Fast Simulation Tool for Showers in High Granularity Calorimeters based on Deep Generative Models** Ph.D. thesis Universität Hamburg

[156] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn 2021 **GANplifying event samples** *SciPost Physics* **10** doi: 10.21468/scipostphys.10.6.139

[157] S. Bieringer, A. Butter, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, B. Nachman, T. Plehn and M. Trabs 2022 **Calomplification – The Power of Generative Calorimeter Models** *JINST* **17** P09028. e-Print: 2202.07352 doi: 10.1088/1748-0221/17/09/P09028

[158] S. Bieringer, S. Diefenbacher, G. Kasieczka and M. Trabs 2024 **Calibrating Bayesian Generative Machine Learning for Bayesiamplification**. e-Print: 2408.00838 URL: https://arxiv.org/abs/2408.00838

[159] I. Goodfellow, Y. Bengio and A. Courville 2016 **Deep Learning** (MIT Press) http://www.deeplearningbook.org

[160] C. M. Bishop and H. Bishop 2024 **Deep Learning: Foundations and Concepts** (Springer)

[161] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher and D. J. Schwab 2019 **A high-bias, low-variance introduction to Machine Learning for physicists** *Physics Reports* **810** 1–124 doi: 10.1016/j.physrep.2019.03.001

[162] M. Erdmann, J. Glombitza, G. Kasieczka and U. Klemradt 2021 **Deep Learning for Physics Research** (WORLD SCIENTIFIC). e-Print: https://worldscientific.com/doi/pdf/10.1142/12294 doi: 10.1142/12294 URL: http://deeplearningphysics.org

[163] D. E. Rumelhart, G. E. Hinton and R. J. Williams 1986 **Learning representations by back-propagating errors** *Nature* **323** 533–536 doi: 10.1038/323533a0

[164] L. Bottou 2012 **Stochastic Gradient Descent Tricks** (Berlin, Heidelberg: Springer Berlin Heidelberg) pp 421–436 ISBN 978-3-642-35289-8 doi: 10.1007/978-3-642-35289-8_25

[165] J. Duchi, E. Hazan and Y. Singer 2011 **Adaptive Subgradient Methods for Online Learning and Stochastic Optimization** *Journal of Machine Learning Research* **12** 2121–2159

[166] D. P. Kingma and J. Ba 2017 **Adam: A Method for Stochastic Optimization**. e-Print: 1412.6980

[167] I. Loshchilov and F. Hutter 2017 **SGDR: Stochastic Gradient Descent with Warm Restarts**. e-Print: 1608.03983 URL: https://arxiv.org/abs/1608.03983

[168] M. A. Nielsen 2015 **Neural networks and deep learning** (Determination Press) available at: http://neuralnetworksanddeeplearning.com

[169] H. Mhaskar, Q. Liao and T. Poggio 2016 **Learning Functions: When Is Deep Better Than Shallow**. e-Print: 1603.00988 URL: https://arxiv.org/abs/1603.00988

[170] K. He, X. Zhang, S. Ren and J. Sun 2015 **Deep Residual Learning for Image Recognition** arXiv URL: https://arxiv.org/abs/1512.03385 doi: 10.48550/ARXIV.1512.03385

[171] O. Ronneberger, P. Fischer and T. Brox 2015 **U-Net: Convolutional Networks for Biomedical Image Segmentation**. e-Print: 1505.04597

[172] D. E. Rumelhart and D. Zipser 1985 **Feature discovery by competitive learning** *Cognitive Science* **9** 75–112 doi: https://doi.org/10.1016/S0364-0213(85)80010-0

[173] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf *et al.* 2019 **PyTorch: An Imperative Style, High-Performance Deep Learning Library**. e-Print: 1912.01703

[174] Y. LeCun and Y. Bengio 1998 **Convolutional networks for images, speech, and time series** (Cambridge, MA, USA: MIT Press) pp 255–258 ISBN 0262511029

[175] M. D. Zeiler, D. Krishnan, G. W. Taylor and R. Fergus 2010 **Deconvolutional networks** *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* pp 2528–2535 doi: 10.1109/CVPR.2010.5539957

[176] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov and A. Smola 2017 **Deep Sets** arXiv URL: https://arxiv.org/abs/1703.06114 doi: 10.48550/ARXIV.1703.06114

[177] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song *et al.* 2018 **Relational inductive biases, deep learning, and graph networks**. e-Print: 1806.01261

[178] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl 2017 **Neural Message Passing for Quantum Chemistry** arXiv URL: https://arxiv.org/abs/1704.01212 doi: 10.48550/ARXIV.1704.01212

[179] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio 2018 **Graph Attention Networks** *International Conference on Learning Representations*

[180] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin 2023 **Attention Is All You Need**. e-Print: 1706.03762

[181] P. T. Komiske, E. M. Metodiev and J. Thaler 2019 **Energy flow networks: deep sets for particle jets** *Journal of High Energy Physics* **2019** doi: 10.1007/jhep01(2019)121

[182] L. Ruthotto and E. Haber 2021 **An Introduction to Deep Generative Modeling**. e-Print: 2103.05180 URL: https://arxiv.org/abs/2103.05180

[183] J. M. Tomczak 2022 **Deep Generative Modeling** (Springer Cham)

[184] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio 2014 **Generative Adversarial Networks**. e-Print: 1406.2661 URL: https://arxiv.org/abs/1406.2661

[185] M. Mirza and S. Osindero 2014 **Conditional Generative Adversarial Nets**. e-Print: 1411.1784 URL: https://arxiv.org/abs/1411.1784

[186] A. Sauer, K. Schwarz and A. Geiger 2022 **StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets** vol abs/2201.00273 URL: https://arxiv.org/abs/2201.00273

[187] M. Kang, J.-Y. Zhu, R. Zhang, J. Park, E. Shechtman, S. Paris and T. Park 2023 **Scaling up GANs for Text-to-Image Synthesis**. e-Print: 2303.05511 URL: https://arxiv.org/abs/2303.05511

[188] A. Xu, S. Han, X. Ju and H. Wang 2024 **Generative machine learning for detector response modeling with a conditional normalizing flow** *JINST* **19** P02003. e-Print: 2303.10148 doi: 10.1088/1748-0221/19/02/P02003

[189] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang and S. P. Smolley 2016 **Least Squares Generative Adversarial Networks** arXiv URL: https://arxiv.org/abs/1611.04076 doi: 10.48550/ARXIV.1611.04076

[190] M. Arjovsky, S. Chintala and L. Bottou 2017 **Wasserstein GAN**. e-Print: 1701.07875

[191] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin and A. Courville 2017 **Improved Training of Wasserstein GANs**. e-Print: 1704.00028

[192] H. Bourlard and Y. Kamp 1988 **Auto-association by multilayer perceptrons and singular value decomposition** *Biological Cybernetics* **59** 291–294 doi: 10.1007/BF00332918

[193] D. P. Kingma and M. Welling 2022 **Auto-Encoding Variational Bayes**. e-Print: 1312.6114

[194] D. P. Kingma and M. Welling 2019 **An Introduction to Variational Autoencoders** *Foundations and Trends® in Machine Learning* **12** 307–392 doi: 10.1561/2200000056

[195] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed and A. Lerchner 2017 **beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework** *International Conference on Learning Representations* URL: https://openreview.net/forum?id=Sy2fzU9gl

[196] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow and B. Frey 2016 **Adversarial Autoencoders** e-Print: 1511.05644

[197] N. Tishby, F. C. Pereira and W. Bialek 2000 **The information bottleneck method**. e-Print: physics/0004057 URL: https://arxiv.org/abs/physics/0004057

[198] N. Tishby and N. Zaslavsky 2015 **Deep Learning and the Information Bottleneck Principle**. e-Print: 1503.02406 URL: https://arxiv.org/abs/1503.02406

[199] S. Voloshynovskiy, M. Kondah, S. Rezaeifar, O. Taran, T. Holotyak and D. J. Rezende 2019 **Information bottleneck through variational glasses**. e-Print: 1912.00830

[200] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf and A. J. Smola 2008 **A Kernel Method for the Two-Sample Problem** *CoRR* e-Print: 0805.2368

[201] S. Zhao, J. Song and S. Ermon 2018 **InfoVAE: Information Maximizing Variational Autoencoders**. e-Print: 1706.02262 URL: https://arxiv.org/abs/1706.02262

[202] A. B. L. Larsen, S. K. Sønderby, H. Larochelle and O. Winther 2016 **Autoencoding beyond pixels using a learned similarity metric**. e-Print: 1512.09300

[203] L. Dinh, D. Krueger and Y. Bengio 2015 **NICE: Non-linear Independent Components Estimation**. e-Print: 1410.8516 URL: https://arxiv.org/abs/1410.8516

[204] I. Kobyzev, S. J. Prince and M. A. Brubaker 2021 **Normalizing Flows: An Introduction and Review of Current Methods** *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43** 3964–3979 doi: 10.1109/tpami.2020.2992934

[205] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed and B. Lakshminarayanan 2021 **Normalizing Flows for Probabilistic Modeling and Inference**. e-Print: 1912.02762 URL: https://arxiv.org/abs/1912.02762

[206] L. Dinh, J. Sohl-Dickstein and S. Bengio 2017 **Density estimation using Real NVP**. e-Print: 1605.08803

[207] R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud 2019 **Neural Ordinary Differential Equations**. e-Print: 1806.07366 doi: 10.48550/arxiv.1806.07366

[208] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze and E. Mishchenko 1962 **The Mathematical Theory of Optimal Processes** (Interscience Publishers John Wiley & Sons, Inc.) translated from the Russian by K.N. Trirogoff

[209] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever and D. Duvenaud 2018 **FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models**. e-Print: 1810.01367 URL: https://arxiv.org/abs/1810.01367

[210] X. Liu, C. Gong and Q. Liu 2022 **Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow**. e-Print: 2209.03003 doi: 10.48550/arxiv.2209.03003

[211] M. S. Albergo and E. Vanden-Eijnden 2023 **Building Normalizing Flows with Stochastic Interpolants**. e-Print: 2209.15571 doi: 10.48550/arxiv.2209.15571

[212] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel and M. Le 2023 **Flow Matching for Generative Modeling**. e-Print: 2210.02747 doi: 10.48550/arxiv.2210.02747

[213] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan and S. Ganguli 2015 **Deep Unsupervised Learning using Nonequilibrium Thermodynamics**. e-Print: 1503.03585

[214] J. Ho, A. Jain and P. Abbeel 2020 **Denoising diffusion probabilistic models** *Proceedings of the 34th International Conference on Neural Information Processing Systems* NIPS '20 (Red Hook, NY, USA: Curran Associates Inc.) ISBN 9781713829546

[215] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon and B. Poole 2021 **Score-Based Generative Modeling through Stochastic Differential Equations**. e-Print: 2011.13456

[216] P. Dhariwal and A. Nichol 2021 **Diffusion Models Beat GANs on Image Synthesis**. e-Print: 2105.05233

[217] Y. Song and S. Ermon 2020 **Generative Modeling by Estimating Gradients of the Data Distribution**. e-Print: 1907.05600

[218] T. Salimans and J. Ho 2022 **Progressive Distillation for Fast Sampling of Diffusion Models**. e-Print: 2202.00512

[219] Y. Song, P. Dhariwal, M. Chen and I. Sutskever 2023 **Consistency Models**. e-Print: 2303.01469

[220] A. Hyvärinen 2005 **Estimation of Non-Normalized Statistical Models by Score Matching** *J. Mach. Learn. Res.* **6** 695–709

[221] P. Vincent 2011 **A Connection Between Score Matching and Denoising Autoencoders** *Neural Computation* **23** 1661–1674 doi: 10.1162/NECO_a_00142

[222] P. E. Kloeden and E. Platen 1992 **Numerical Solution of Stochastic Differential Equations** (Springer Berlin) doi: 10.1007/978-3-662-12616-5

[223] T. Karras, M. Aittala, T. Aila and S. Laine 2022 **Elucidating the Design Space of Diffusion-Based Generative Models**. e-Print: 2206.00364

[224] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn *et al.* 2024 **Scaling Rectified Flow Transformers for High-Resolution Image Synthesis**. e-Print: 2403.03206 URL: https://arxiv.org/abs/2403.03206

[225] Y. Song and P. Dhariwal 2023 **Improved Techniques for Training Consistency Models**. e-Print: 2310.14189 URL: https://arxiv.org/abs/2310.14189

[226] N. Tishby, F. C. Pereira and W. Bialek 2000 **The information bottleneck method** arXiv preprint physics/0004057 [physics.data-an]

[227] The iLCSoft Group 2016 **iLCSoft Project Page** [Online; accessed 10-September-2022] URL: `https://github.com/iLCSoft`

[228] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio 2014 **Generative Adversarial Networks**. e-Print: `1406.2661`

[229] J. L. Ba, J. R. Kiros and G. E. Hinton 2016 **Layer Normalization** e-Print: `1607.06450`

[230] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.* 2013 **Rectifier nonlinearities improve neural network acoustic models** *Proc. icml* vol 30 (Citeseer) p 3 URL: `http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf`

[231] T. D. Kulkarni, W. Whitney, P. Kohli and J. B. Tenenbaum 2015 **Deep Convolutional Inverse Graphics Network**. e-Print: `1503.03167`

[232] C. E. Shannon 1948 **A mathematical theory of communication.** Bell Syst. Tech. J. **27**, 379 URL: `http://dblp.uni-trier.de/db/journals/bstj/bstj27.html#Shannon48`

[233] S. Kullback 1959 **Information Theory and Statistics** (New York: Wiley)

[234] D. M. Endres and J. E. Schindelin 2003 **A new metric for probability distributions** *IEEE Transactions on Information Theory* **49** 1858–1860

[235] D. Freedman, R. Pisani and R. Purves 2007 **Statistics (international student edition)** *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*

[236] J. Feydy, T. Séjourné, F.-X. Vialard, S.-i. Amari, A. Trouve and G. Peyré 2019 **Interpolating between Optimal Transport and MMD using Sinkhorn Divergences** *The 22nd International Conference on Artificial Intelligence and Statistics* pp 2681–2690

[237] R. L. Dobrushin 1970 **Prescribing a System of Random Variables by Conditional Distributions** *Theory of Probability & Its Applications* **15** 458–486. e-Print: `https://doi.org/10.1137/1115049` doi: `10.1137/1115049`

[238] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett *et al.* 2020 **SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python** *Nature Methods* **17** 261–272 doi: `10.1038/s41592-019-0686-2`

[239] J. Feydy 2020 **Geometric data analysis, beyond convolutions** Ph.D. thesis Université Paris-Saclay

[240] R. Kansal, A. Li, J. Duarte, N. Chernyavskaya, M. Pierini, B. Orzari and T. Tomei 2023 **Evaluating generative models in high energy physics** *Physical Review D* **107** doi: `10.1103/physrevd.107.076017`

[241] R. Das, L. Favaro, T. Heimel, C. Krause, T. Plehn and D. Shih 2023 **How to Understand Limitations of Generative Networks**. e-Print: 2305.16774

[242] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter 2018 **GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium**. e-Print: 1706.08500

[243] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich 2014 **Going Deeper with Convolutions**. e-Print: 1409.4842

[244] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei 2009 **ImageNet: A large-scale hierarchical image database** *2009 IEEE Conference on Computer Vision and Pattern Recognition* pp 248–255 doi: 10.1109/CVPR.2009.5206848

[245] H. Qu and L. Gouskos 2020 **Jet tagging via particle clouds** *Physical Review D* **101** doi: 10.1103/physrevd.101.056019

[246] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller and W. Samek 2015 **On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation** *PLoS ONE* **10** e0130140

[247] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders and K.-R. Müller 2021 **Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications** *Proceedings of the IEEE* **109** 247–278 doi: 10.1109/JPROC.2021.3060483

[248] S. Schnake, D. Krücker and K. Borras 2022 **Generating Calorimeter Showers as Point Clouds** URL: https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_77.pdf

[249] F. T. Acosta, V. Mikuni, B. Nachman, M. Arratia, B. Karki, R. Milton, P. Karande and A. Angerami 2023 **Comparison of Point Cloud and Image-based Models for Calorimeter Fast Simulation**. e-Print: 2307.04780

[250] J. Liu, A. Ghosh, D. Smith, P. Baldi and D. Whiteson 2023 **Generalizing to new geometries with Geometry-Aware Autoregressive Models (GAAMs) for fast calorimeter simulation** *Journal of Instrumentation* **18** P11003 doi: 10.1088/1748-0221/18/11/p11003

[251] S. Luo and W. Hu 2021 **Diffusion Probabilistic Models for 3D Point Cloud Generation**. e-Print: 2103.01458

[252] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever and D. Duvenaud 2018 **FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models**. e-Print: 1810.01367

[253] A. Gu, K. Goel and C. Ré 2022 **Efficiently Modeling Long Sequences with Structured State Spaces**. e-Print: 2111.00396

[254] K. He, X. Zhang, S. Ren and J. Sun 2015 **Deep Residual Learning for Image Recognition**. e-Print: 1512.03385

[255] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios 2019 **Neural Spline Flows**. e-Print: 1906.04032

[256] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios 2020 **nflows: normalizing flows in PyTorch** Zenodo URL: https://doi.org/10.5281/zenodo.4296287 doi: 10.5281/zenodo.4296287

[257] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall and N. D. Goodman 2018 **Pyro: Deep Universal Probabilistic Programming** *Journal of Machine Learning Research*

[258] V. Mikuni, B. Nachman and M. Pettee 2023 **Fast point cloud generation with diffusion models in high energy physics** *Phys. Rev. D* **108** 036025. e-Print: 2304.01266 doi: 10.1103/PhysRevD.108.036025

[259] T. Dao, D. Y. Fu, S. Ermon, A. Rudra and C. Ré 2022 **FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness**. e-Print: 2205.14135

[260] E. Buhmann, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger, M. Mozzanica and L. Valente 2024 **It's about time: a point cloud generative model for the CMS High granularity calorimeter** *in preparation*

[261] H. Qu, C. Li and S. Qian 2022 **JetClass: A Large-Scale Dataset for Deep Learning in Jet Physics** Zenodo URL: https://doi.org/10.5281/zenodo.6619768 doi: 10.5281/zenodo.6619768

[262] M. Leigh, D. Sengupta, G. Quétant, J. A. Raine, K. Zoch and T. Golling 2024 **PC-JeDi: Diffusion for particle cloud generation in high energy physics** *SciPost Phys.* **16** 018. e-Print: 2303.05376 doi: 10.21468/SciPostPhys.16.1.018

[263] M. Leigh, D. Sengupta, J. A. Raine, G. Quétant and T. Golling 2024 **Faster diffusion model with improved quality for particle cloud generation** *Physical Review D* **109** doi: 10.1103/physrevd.109.012010

[264] R. Kansal *et al.* 2022 **JetNet** Zenodo doi: 10.5281/zenodo.6975118

[265] R. Kansal *et al.* 2022 **JetNet150** Zenodo doi: 10.5281/zenodo.6302240

[266] J. Song, C. Meng and S. Ermon 2022 **Denoising Diffusion Implicit Models**. e-Print: 2010.02502

[267] B. Käch and I. Melzer-Pellmann 2023 **Attention to Mean-Fields for Particle Cloud Generation** e-Print: 2305.15254

[268] B. Käch, I. Melzer-Pellmann and D. Krücker 2024 **Pay Attention To Mean Fields For Point Cloud Generation** e-Print: 2408.04997

[269] B. Käch, D. Krücker, I. Melzer-Pellmann, M. Scham, S. Schnake and A. Verney-Provatas 2022 **JetFlow: Generating Jets with Conditioned and Mass Constrained Normalising Flows** e-Print: 2211.13630

[270] E. Coleman, M. Freytsis, A. Hinzmann, M. Narain, J. Thaler, N. Tran and C. Vernieri 2018 **The importance of calorimetry for highly-boosted jet substructure** *JINST* **13** T01003 doi: 10.1088/1748-0221/13/01/T01003

[271] M. Pierini, J. M. Duarte, N. Tran and M. Freytsis 2020 **HLS4ML LHC Jet dataset (30 particles)** Zenodo URL: https://doi.org/10.5281/zenodo.3601436 doi: 10.5281/zenodo.3601436

[272] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli and M. Zaro 2014 **The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations** *JHEP* **07** 079. e-Print: 1405.0301 doi: 10.1007/JHEP07(2014)079

[273] P. Skands, S. Carrazza and J. Rojo 2014 **Tuning PYTHIA 8.1: the Monash 2013 Tune** *Eur. Phys. J. C* **74** 3024. e-Print: 1404.5630 doi: 10.1140/epjc/s10052-014-3024-y

[274] E. A. Moreno, O. Cerri, J. M. Duarte, H. B. Newman, T. Q. Nguyen, A. Periwal, M. Pierini, A. Serikova, M. Spiropulu and J.-R. Vlimant 2020 **JEDI-net: a jet identification algorithm based on interaction networks** *The European Physical Journal C* **80** doi: 10.1140/epjc/s10052-020-7608-4

[275] M. Cacciari, G. P. Salam and G. Soyez 2008 **The anti-kt jet clustering algorithm** *JHEP* **04** 063 doi: 10.1088/1126-6708/2008/04/063

[276] M. Cacciari, G. P. Salam and G. Soyez 2012 **FastJet User Manual** *Eur. Phys. J. C* **72** 1896 doi: 10.1140/epjc/s10052-012-1896-2

[277] S. R. Qasim, J. Kieseler, Y. Iiyama and M. Pierini 2019 **Learning representations of irregular particle-detector geometry with distance-weighted graph networks** *The European Physical Journal C* **79** doi: 10.1140/epjc/s10052-019-7113-9

[278] T. Salimans and D. P. Kingma 2016 **Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks**. e-Print: 1602.07868

[279] G. J. Székely and M. L. Rizzo 2009 **Brownian distance covariance** *The Annals of Applied Statistics* **3** 1236 – 1265 doi: 10.1214/09-AOAS312

[280] Y. Song and S. Ermon 2020 **Improved Techniques for Training Score-Based Generative Models**. e-Print: 2006.09011

[281] A. Butter, N. Huetsch, S. P. Schweitzer, T. Plehn, P. Sorrenson and J. Spinner 2023 **Jet Diffusion versus JetGPT – Modern Networks for the LHC**. e-Print: 2305.10475

[282] S. Diefenbacher, V. Mikuni and B. Nachman 2023 **Refining Fast Calorimeter Simulations with a Schrödinger Bridge**. e-Print: 2308.12339

[283] I. Loshchilov and F. Hutter 2019 **Decoupled Weight Decay Regularization** e-Print: 1711.05101 doi: 10.48550/arXiv.1711.05101

[284] R. T. Q. Chen 2018 **torchdiffeq** URL: https://github.com/rtqichen/torchdiffeq

[285] A. van den Oord and B. Schrauwen 2014 **Factoring Variations in Natural Images with Deep Gaussian Mixture Models** *Advances in Neural Information Processing Systems* vol 27 edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence and K. Weinberger (Curran Associates, Inc.)

[286] G. Kasieczka, B. Nachman, D. Shih, O. Amram, A. Andreassen, K. Benkendorfer, B. Bortolato, G. Brooijmans, F. Canelli, J. H. Collins, B. Dai, F. F. De Freitas *et al.* 2021 **The LHC Olympics 2020 a community challenge for anomaly detection in high energy physics** *Reports on Progress in Physics* **84** 124201 doi: 10.1088/1361-6633/ac36b9

[287] A. Hallin, J. Isaacson, G. Kasieczka, C. Krause, B. Nachman, T. Quadfasel, M. Schlaffer, D. Shih and M. Sommerhalder 2021 **Classifying Anomalies THrough Outer Density Estimation (CATHODE)** e-Print: 2109.00546

[288] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer 2022 **High-Resolution Image Synthesis with Latent Diffusion Models**. e-Print: 2112.10752 URL: https://arxiv.org/abs/2112.10752

[289] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji and S. Ermon 2024 **Consistency Trajectory Models: Learning Probability Flow ODE Trajectory of Diffusion**. e-Print: 2310.02279 URL: https://arxiv.org/abs/2310.02279

Eidesstattliche Versicherung

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.

02.02.2025

Datum

Unterschrift der Doktorandin/des Doktoranden