

# Topics in Generative Modeling of Particle Physics Data

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

vorgelegt von  
Sascha Daniel Diefenbacher

Hamburg  
2022



Gutachter/innen der Dissertation:	- Prof. Dr. Gregor Kasieczka - Dr. Frank Gaede
Zusammensetzung der Prüfungskommission:	- Prof. Dr. Gregor Kasieczka - Prof. Dr. Freya Blekman - Prof. Dr. Marcus Brüggem - Prof. Dr. Günter H. W. Sigl - Dr. Frank Gaede
Vorsitzende/r der Prüfungskommission:	- Prof. Dr. Günter H. W. Sigl
Datum der Disputation:	11.01.2023
Vorsitzender Fach-Promotionsausschusses PHYSIK:	Prof. Dr. Günter H. W. Sigl
Leiter des Fachbereichs PHYSIK:	Prof. Dr. Wolfgang J. Parak
Dekan der Fakultät MIN:	Prof. Dr.-Ing. Norbert Ritter



# Zusammenfassung

Messungen in Teilchenphysik-Experimenten resultieren in gewaltigen Datenmengen. Das Festhalten dieser Daten braucht auf der einen Seite eine enorme Menge an Speicherplatz, und auf der anderen Seite ist eine ähnlich große Menge an Simulationsdaten nötig, um die Messungen zu analysieren. Ausreichend Rechenressourcen für diese beiden Anwendungen bereitzustellen, wird kontinuierlich schwieriger. Diese Arbeit untersucht daher, wie generative Anwendungen von maschinellem Lernen genutzt werden können, um diesem Problem beizukommen.

Wir demonstrieren zuerst, dass ein generatives Modell tatsächlich in der Lage ist, eine größere Anzahl neuer Datenpunkte zu generieren, als benutzt wurden, um das Modell zu trainieren. Dieses Ergebnis stellt den Grundbaustein für die Anwendung von generativen Modellen für schnelle Simulationen dar. Anschließend daran präsentieren wir, wie drei generativen Modelle – ein GAN, ein WGAN und ein BIB-AE – für die schnelle Simulation von Photon-Kaskaden in dem hoch-granularen elektromagnetischen Kalorimeter des geplanten International Large Detectors verwendet werden können. Des Weiteren zeigen wir, wie die WGAN und BIB-AE Modelle erweitert werden können, um die Simulation von Pion-Kaskaden in einem hadronischen Kalorimeter zu ermöglichen. Sowohl für Photonen, als auch Pionen, zeigen wir, dass die generativen Modelle die Ergebnisse klassischer Simulationsmethoden gut nachahmen können, und dabei signifikant weniger Zeit brauchen, um die Kalorimeter Kaskaden zu simulieren. Darüber hinaus präsentieren wir die Ergebnisse des ersten Trainings eines generativen Modells auf echten Messdaten. Dabei zeigen wir, dass ein so trainiertes Modell eine ähnliche Präzision erreichen kann wie klassische Simulationsmethoden und gleichzeitig wesentlich weniger Simulationszeit und Rechenleistung in Anspruch nimmt. Schlussendlich stellen wir ein online trainiertes generatives Modell vor, welches in der Lage ist Informationen aus Bereichen zu sammeln, die von momentanen Trigger Systemen verworfen werden. Somit könnte ein solches Modell benutzt werden, um den Speicherplatzbedarf moderner Teilchenphysik-Experimente zu reduzieren.



# Abstract

The large amount of data collected by current and future particle physics experiments requires both a large amount of space to store the recorded data and a large amount of simulated data to analyze. This presents a significant strain on the available computational resources. This work explores the use of generative machine learning models to address these challenges.

We initially demonstrate the ability of a generative model to generate more data points than it was trained on, thereby showing generative models are a viable approach for fast simulation. Building on this, we demonstrate the use of three generative networks, a GAN, a WGAN, and a BIB-AE with Post Processor, for the fast simulation of photon showers in a highly granular electromagnetic calorimeter, designed for the International Large Detector. We further show how the WGAN and BIB-AE models can be extended to simulate pion showers in a hadronic calorimeter with a high degree of accuracy, while significantly reducing the needed per-shower simulation time. Notably, we also present the first results for a generative model trained on measurement data in particle physics and show that a BIB-AE model, trained on testbeam data, can reach a precision similar to classical simulation tools while providing a significant speedup. Finally, we address the challenge of having limited storage space by presenting a proposal for an online trained generative model. We show that this model can act as a scouting tool for regions currently ignored by trigger setups and be used to extract potential new-physics signals from these regions without requiring additional storage space.





# Contents

<b>Preface</b>	<b>1</b>
<b>Introduction</b>	<b>3</b>
<b>I Theory and Methodology</b>	<b>7</b>
<b>1 Particle Physics</b>	<b>9</b>
1.1 The Standard Model of Particle Physics . . . . .	9
1.2 Physics Beyond the Standard Model . . . . .	13
1.3 Collider Experiments . . . . .	15
1.4 Linear Lepton Colliders . . . . .	18
1.5 Precision Measurements at Lepton Colliders . . . . .	20
1.6 Monte Carlo Simulation . . . . .	23
<b>2 Calorimetry</b>	<b>27</b>
2.1 Electromagnetic Showers . . . . .	27
2.2 Heavy Charged Particles . . . . .	31
2.3 Hadronic Showers . . . . .	33
2.4 Calorimeter Types . . . . .	34
2.5 Calorimeter and Shower Simulation . . . . .	37
2.6 ILD Calorimeters . . . . .	39
<b>3 Machine Learning</b>	<b>41</b>
3.1 Gradient Descent . . . . .	42
3.2 Optimizers . . . . .	43
3.3 Loss Functions . . . . .	45
3.4 Challenges of Machine Learning . . . . .	46
3.5 Neural Networks . . . . .	47
3.6 Activation Functions . . . . .	53
<b>4 Generative Models</b>	<b>55</b>
4.1 Challenges of Generative Models . . . . .	55
4.2 Generative Adversarial Networks . . . . .	57
4.3 Wasserstein GANs . . . . .	58
4.4 Autoencoders . . . . .	61

4.5	Variational Autoencoders . . . . .	63
4.6	Adversarial Autoencoders . . . . .	68
4.7	The BIB-AE . . . . .	69
4.8	Normalizing Flows . . . . .	74
4.9	Conditional Generative Models . . . . .	79
<b>II</b>	<b>Generative Models in Particle Physics</b>	<b>81</b>
<b>5</b>	<b>GANplification</b>	<b>83</b>
5.1	1-Dimensional Data . . . . .	84
5.2	2-Dimensional Data . . . . .	89
5.3	Multi-Dimensional Data . . . . .	91
5.4	Conclusion . . . . .	93
<b>6</b>	<b>Photon Shower Generation</b>	<b>95</b>
6.1	Photon Data Set . . . . .	95
6.2	GAN and WGAN Models . . . . .	97
6.3	BIB-AE Model . . . . .	99
6.4	Results . . . . .	102
6.5	Conclusion . . . . .	111
<b>7</b>	<b>Pion Shower Generation</b>	<b>113</b>
7.1	Pion Data Set . . . . .	113
7.2	WGAN Model . . . . .	115
7.3	BIB-AE Model . . . . .	115
7.4	Results . . . . .	121
7.5	Conclusion . . . . .	128
<b>8</b>	<b>CALICE Testbeam Data Generation</b>	<b>129</b>
8.1	Testbeam Data Set . . . . .	130
8.2	BIB-AE Model . . . . .	131
8.3	Results . . . . .	132
8.4	Conclusion . . . . .	141
<b>9</b>	<b>OnlineFlow</b>	<b>143</b>
9.1	Online Training . . . . .	144
9.2	Bumphunt Data Set . . . . .	146
9.3	Anomaly Detection Data Set . . . . .	152
9.4	Conclusion . . . . .	155
<b>10</b>	<b>Summary and Outlook</b>	<b>157</b>
	<b>Acknowledgments</b>	<b>161</b>
	<b>Bibliography</b>	<b>163</b>
<b>A</b>	<b>Neural Network Architectures</b>	<b>179</b>

# Preface

The findings presented in this thesis are the result of research conducted in collaboration with other researchers, in the time between 2019 and 2022 at the Institut für Experimentalphysik of the Universität Hamburg. The results shown in Chapters 5, 6, 7, and 9 have previously been published as

- [1]: A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, and T. Plehn 2021 **GANplifying event samples** *SciPost Phys.* **10** 139. 10.21468/SciPostPhys.10.6.139
- [2]: 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
- [3]: 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
- [4]: A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, T. Plehn, D. Shih and R. Winterhalder 2022 **Ephemeral Learning — Augmenting Triggers with Online-Trained Normalizing Flows** 2202.09375

Additionally the author has been involved in the following publications during this research period:

- [5]: S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman, and D. Shih 2020 **DCTR-GAN: Improving the Precision of Generative Models with Reweighting** *JINST* **15** P11004. 10.1088/1748-0221/15/11/P11004
- [6]: 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
- [7]: S. Bieringer, A. Butter, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, B. Nachman, Benjamin. T. Plehn, and M. Trabs 2022 **Calomplification – the power of generative calorimeter models** *JINST* **17** P09028. 10.1088/1748-0221/17/09/P09028



# Introduction

It is the fundamental goal of physics to understand the nature of the universe. The study of particle physics furthers this goal by investigating elementary particles and the laws that govern their interactions. The standard model of particle physics [8–11] represents an incredible achievement in this regard. It manages to describe the electromagnetic, strong, and weak interactions between particles with unprecedented accuracy and has been confirmed through experimental results again and again. With the discovery of the Higgs boson [12–14] by the ATLAS and CMS collaborations in 2012 [15, 16] one of the last missing pieces of the standard model was found, and the model was made self-consistent.

However, despite its experimental accuracy and theoretical consistency, the standard model is incomplete. Several observed phenomena, such as the existence of dark matter [17–20], apparent neutrino masses [21], and indications of flavor anomalies [22, 23] are not accounted for in the standard model.

Current collider experiments, such as the Large Hadron Collider (LHC) [24] have yet to discover clear evidence of physics beyond the standard model, indicating that further particle physics discoveries may require significantly higher energies or improved measurement precision. Several proposed approaches exist, that aim to provide this increased precision, ranging from upgrades to current experiments, such as the High-Luminosity LHC [25], to entirely new experiments, such as linear lepton colliders like the International Linear Collider (ILC) [26] or the Compact Linear Collider (CLIC) [27]. One feature shared by all these approaches is their high luminosity, giving them the ability to produce vast amounts of measurement data.

In order to analyze the recorded data, it needs to be compared to theoretical predictions. However, this comparison is highly non-trivial, as the subtle effects that different theoretical models would have on the measurement data have to be modeled. Such modelings are exceedingly complex and have to rely on Monte Carlo (MC) simulations. A full particle physics simulation starts with event generation, where the initial collision, or hard process, is modeled. Then, hadronization is simulated for all particles resulting from the initial collision, and finally, the exact response of the detector to all produced particles is modeled. Full computation of this chain has to be performed for every simulated event and requires significant computational time and resources.

Most notably, the simulation of calorimeter showers takes up a significant fraction of the total simulation resources [28]. Calorimeters are large detector volumes, designed to absorb and measure the energy of particles. When certain particles interact with a calorimeter, they cause a cascade of further particles in the calorimeter, known as a shower. Classical simulation of calorimeter showers is performed using GEANT4 [29], a state-of-the-art simulation package that models each individual particle in a shower. This process takes a significant amount of time and makes classical calorimeter simulation a bottleneck in the simulation chain.

For the comparison between MC simulated data and measurement to be sufficiently precise, one aims to use enough simulated data to at least match, ideally exceed, the amount of recorded data. A move to higher collider luminosities, therefore, leads to an increased demand for MC simulated data, which, in turn, requires an increased computational budget. Under current projections, generating the required MC data to keep up with future collider experiments will soon require more computational resources than are readily available [30].

Simultaneous with the recent work towards new and more precise colliders, machine learning (ML) methods have begun to see an increased application in particle physics. The concepts of ML have existed for several decades. However, only through recent advances in computer science, most notably in the form of highly sophisticated optimization algorithms, the development of parallel computing architectures such as graphical processing units (GPUs), and the greater availability of large data sets, has ML moved into the realm of practical application.

ML applications can be split into three groups, supervised methods that operate on labeled data, unsupervised methods that work on unlabeled data, and reinforcement learning, which aims to learn decision making. In particle physics, supervised ML methods have become established for particle identification and jet flavor tagging tasks [31–33], and several unsupervised applications are under development, such as using anomaly detection setups for model-independent physics searches [34].

Generative models are an exciting subset of unsupervised ML methods. These models aim to learn the underlying distribution of a data set and use this learned distribution to generate new data. Recently generative models have been in the public spotlight due to the emergence of generative art engines, capable of producing high-quality artwork based only on a text prompt.

This makes generative models a promising solution to the problem of escalating simulation costs. Unlike classical simulation tools, generative models do not have to rely on slow methods like MC integration or individual modeling of shower constituents. Therefore, a generative model trained to produce MC simulation data can produce this data significantly faster than classical simulation methods.

There exists a wide range of efforts to use generative models for fast simulation. The most common method utilizes classical simulation tools to generate a set of data, which is then used to train a generative model. The trained model is then used to quickly generate large amounts of simulation data. This method has been applied to event generation [35–41], hadronization [42–45], and detector simulation [46–57]. A notable alternative approach to having a generative model perform the complete simulation is the use of ML methods to improve classical fast simulation methods through reweighting [58–60] or refinement [61].

In addition to providing a significant speedup over classical simulation methods, generative models also offer new possibilities for particle simulation. Notably, they may allow for simulating the entire detector in one step, combining both detector simulation and reconstruction into a single process, or directly training on data in order to achieve an improved accuracy compared to classical methods.

In this work, we explore the application of generative models for the simulation of particle showers in the calorimeters of the International Large Detector (ILD) [59]. The ILD is a proposed detector for the ILC. To enable the high-precision measurements planned for the ILC, the ILD is designed for high resolution and is expected to record large amounts of data. This makes classical simulations of the ILD calorimeters costly and presents a strong incentive for the development of fast ML-based simulation. Most notably we demonstrate the possibility of directly training the

generative model on real data, using testbeam data recorded by the CALICE [62] collaboration with an Analogue Hadronic Calorimeter (AHCAL) [63] prototype.

This work is organized into two parts. In Part I we cover the relevant theoretical basics and underlying methodology. In Chapter 1 a brief summary of the standard model is given, alongside an overview of current and proposed particle physics experiments and detectors. Chapter 2 focuses on the operating principles, construction, and simulation of calorimeter systems for particle detectors. We then present general principles of ML in Chapter 3, before providing a more detailed discussion of generative models in Chapter 4.

Part II presents the application of these principles for fast simulation and novel analysis methods in particle physics. A vital question for the application of generative models as fast simulation is how many points can reasonably be sampled from a generative model. Therefore, in Chapter 5, we initially explore this question using small-scale toy models. Following this, we move to utilize the generative models introduced in Part I for the fast simulation of photon showers in the highly granular electromagnetic sampling calorimeter of the ILD in Chapter 6. We then extend the generative models to significantly more complex pion showers in a hadronic calorimeter in Chapter 7. In Chapter 8 we present the first training of a generative model directly on measurement data in particle physics, using pion testbeam data recorded by the CALICE collaboration. Chapter 9 explores the use of generative models for applications beyond fast simulation, and presents an online trained generative model to try and store information about phase space regions currently discarded by trigger setups. Finally, the obtained results are summarized and an outlook toward future research is given in Chapter 10.





Part I

Theory and Methodology



# Chapter 1

## Particle Physics

The field of particle physics aims to explore the fundamental constituents and underlying forces that make up and shape the universe we inhabit. This Chapter aims to give a brief introduction to the current status of particle physics research. In Section 1.1 the standard model of particle physics is described. Indications for the existence of physics beyond the standard model are discussed in Section 1.2. Section 1.3 covers colliders as a vital tool for particle physics, using the example of the LHC. In Section 1.4 we discuss proposals for new linear lepton colliders and exciting precision measurement opportunities at such linear colliders are outlined in Section 1.5. Finally, the use of MC methods in particle physics is discussed in Section 1.6.

### 1.1 The Standard Model of Particle Physics

The standard model of particle physics [8–11] is one of the most successful theories in all of physics, capable of predicting experimental results with an impressive level of accuracy. The standard model describes the electromagnetic, strong, and weak interactions between fundamental particles as quantum field theories (QFT), which are symmetric under the

$$\text{SU}(3)_C \otimes \text{SU}(2)_L \otimes \text{SU}(1)_Y \tag{1.1.1}$$

gauge symmetry group [64], where each gauge group is related to a fundamental interaction.

#### Fermionic Content

The particles described by the standard model can be grouped into two categories, matter particles, known as fermions and characterized by their spin of  $\frac{1}{2}$ , and particles that mediate interactions, known as bosons, with an integer value spin. The fermionic content of the SM is shown in the leftmost three columns of Figure 1.1 and can be further broken down into leptons, which comprise charged leptons ( $l^-$ ) and lepton neutrinos ( $\nu_l$ ), and quarks, which can again be categorized into up-type quarks ( $q_u$ ) and down-type quarks ( $q_d$ ). Charged leptons and quarks carry an electromagnetic charge, with leptons having a charge of  $-1$  and up-type quarks and down-type quarks having a charge of  $\frac{2}{3}$  and  $-\frac{1}{3}$  respectively. Further, quarks have a color charge, the charge associated with the strong interaction. For both leptons and quarks there exist three

generations that differ only by their mass and flavor, but have otherwise identical quantum numbers. The particles in these generations are electrons ( $e$ ), muons ( $\mu$ ), and taus ( $\tau$ ) for charged leptons and electron-neutrinos ( $\nu_e$ ), muon-neutrinos ( $\nu_\mu$ ), and tau-neutrinos ( $\nu_\tau$ ) for neutral leptons. The three up-type quarks are known as up ( $u$ ), charm ( $c$ ), and top ( $t$ ) quarks, and the three down-type quarks are referred to as down ( $d$ ), strange ( $s$ ), and bottom ( $b$ ) quarks.

For each fermionic particle, there exists a corresponding antiparticle, with the opposite charge and flavor.

Each fermion generation can be decomposed into left- and right-handed components, according to

$$\begin{bmatrix} \nu_l & q_u \\ l^- & q_d \end{bmatrix} = \begin{pmatrix} \nu_l \\ l^- \end{pmatrix}_L, \begin{pmatrix} q_u \\ q_d \end{pmatrix}_L, l^-_R, q_{uR}, q_{dR}, \quad (1.1.2)$$

where the left-handed fermions transform as  $SU(2)_L$  doublets and the right-handed ones as  $SU(2)_L$  singlets. As the neutrino is considered massless in the standard model, there is no right-handed neutrino component  $\nu_{lR}$ . All left-handed particles have a weak iso-spin, the charge corresponding to the weak interaction.

## Fundamental Forces

Each fundamental force in the standard model is mediated by an exchange of bosons. For the electromagnetic interactions, described by quantum electrodynamics (QED), this exchange particle is the massless and chargeless photon  $\gamma$ . Photons couple to the electromagnetic charge, and therefore interact only with charged leptons, quarks, and charged bosons. This also means that photons cannot interact with other photons.

The weak interaction is mediated by three particles, the neutral  $Z$ -boson, and the charged  $W^+$  and  $W^-$  bosons. These bosons only interact with left-handed fermions, as right-handed ones lack the necessary iso-spin. An important note is that the weak interaction does not have to respect the conservation of flavor, such as in cases of a  $t$  quark decaying into a  $b$  quark and a  $W^+$

The mediators of the strong force, described by quantum chromodynamics (QCD), are massless, electromagnetically neutral gluons. Gluons couple to color charge and are themselves color charged, meaning they interact with quarks and other gluons. This gluon-gluon self-interaction results in an observed effect known as color confinement [66], which means that no color charged free particles can exist. Therefore, gluons are always bound in composite states that are, in total, color neutral. Common color neutral states are either groups of 3 quarks, like protons or neutrons, known as baryons, or pairs of quarks and antiquarks, known as mesons, like pions or kaons. Additionally, rare 4 and 5 quarks states, known as tetraquarks [67] and pentaquarks [68] have been observed. Another effect of confinement is that quarks cannot propagate freely, instead creating a spray of colorless hadrons, resulting in a so-called particle jet.

Interactions between particles are described by the standard model Lagrangian, in the form of terms that couple the fields of the corresponding particles involved in the interaction. For example, the interactions between a fermion-antifermion pair,  $\psi\bar{\psi}$ , and the photon gauge field  $A_\mu$  are written as

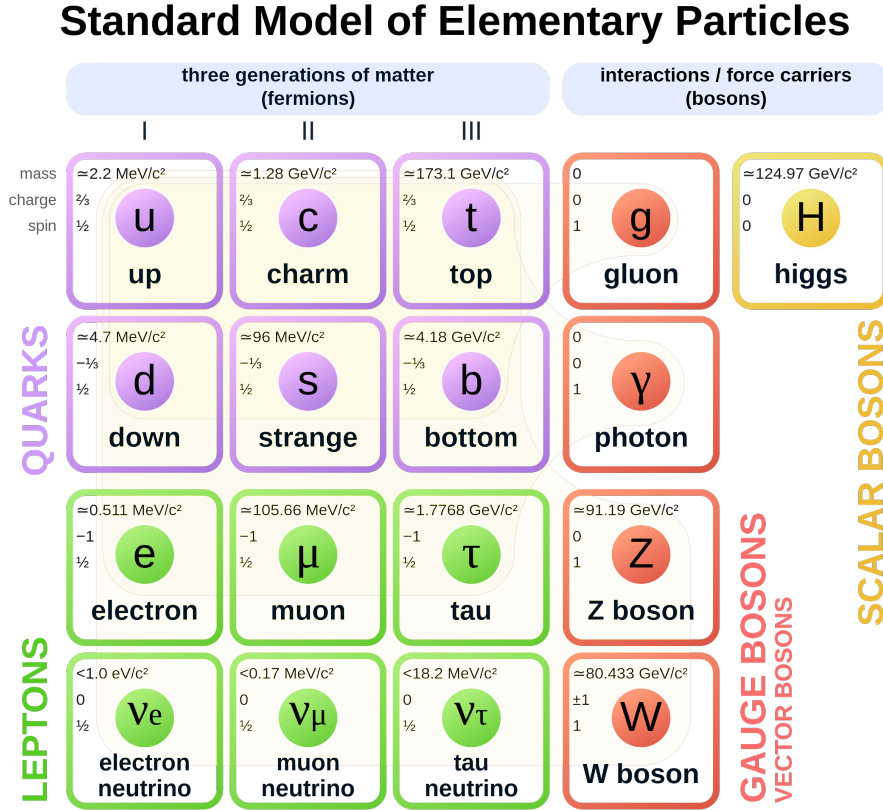


Figure 1.1: Overview of particles in the standard model, listing the mass, electromagnetic charge, and spin of the particles. Figure taken from [65].

$$iq\bar{\psi}\gamma_\mu A_\mu\psi, \quad (1.1.3)$$

where  $q$  is the electromagnetic charge of  $\psi$ , and describes the strength of the coupling. The likelihood of an interaction occurring is described by its cross section  $\sigma$ . For a two-particle to two-particle interaction, this cross section can be calculated as

$$\sigma = \frac{1}{64\pi^2 s} \frac{p_f}{p_i} \int |\mathcal{M}_{fi}|^2 d\Omega, \quad (1.1.4)$$

as described in Reference [66]. In this expression,  $p_f$  and  $p_i$  describe the momenta of the final state and initial state particles in the center-of-mass frame, respectively. The variable  $s$  corresponds to the center-of-mass energy, and  $\mathcal{M}_{fi}$  is the matrix element for the transition from the initial to the final state. Generally speaking, the matrix element depends on the couplings of the particle interaction involved. The matrix element can have an angular dependency and has to be integrated over all possible angles  $\Omega$  in spherical coordinates.

Similarly, one can define the angle-dependent differential cross section as

$$\frac{d\sigma}{d\Omega} = \frac{1}{64\pi^2 s} \frac{p_f}{p_i} \int |\mathcal{M}_{fi}|^2 . \quad (1.1.5)$$

## Higgs Mechanism

The masses of gauge bosons cannot simply be included in the standard model Lagrangian, as they would violate gauge symmetry [66]. Similarly, fermion masses cannot be directly introduced, as the naive fermion mass term

$$m\psi\bar{\psi} = m(\psi_R\bar{\psi}_L + \psi_L\bar{\psi}_R) , \quad (1.1.6)$$

for a fermion field  $\psi$  would couple right-handed fields, that transform as a singlet under  $SU(2)_L$  to left-handed fields, that transform as a doublet, thereby breaking the  $SU(2)_L$  symmetry.

Both of these problems are addressed by the Higgs mechanism [12–14]. It introduces a complex scalar field  $\phi$  with a potential  $V$ , with the form

$$V = \mu^2\phi^\dagger\phi + \lambda(\phi^\dagger\phi)^2 , \quad (1.1.7)$$

where  $\mu^2$  and  $\lambda$  are parameters describing the potential. This potential is itself gauge invariant, however for a negative  $\mu^2$  value the potential has a minimum at  $\phi \neq 0$ . Therefore, the potential has a non-zero vacuum expectation value  $v$ . Due to this expectation value, the field spontaneously breaks local gauge symmetry, giving rise to the boson masses. The resulting masses of the  $W$  and  $Z$  bosons are

$$m_W = g\frac{v}{2} \quad \text{and} \quad m_Z = \sqrt{g^2 + g'^2}\frac{v}{2} , \quad (1.1.8)$$

where  $g^2$  and  $g'^2$  are electroweak coupling constants. Further, the resulting scalar field can, in the right gauge, be written as

$$\phi = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v + h \end{pmatrix} , \quad (1.1.9)$$

where  $h$  can be understood as an excitation of the scalar field, manifesting as a scalar boson known as the Higgs boson. This scalar field can couple to fermions, as the term

$$g_f\psi_R\phi\bar{\psi}_L + g_f\psi_L\phi\bar{\psi}_R , \quad (1.1.10)$$

is invariant under gauge transformation. Here  $g_f$  describes the coupling strength, known as the Yukawa coupling, of a given fermion to the scalar field. This expression can be expanded to

$$g_f v(\psi_R\bar{\psi}_L + \psi_L\bar{\psi}_R) + g_f h(\psi_R\bar{\psi}_L + \psi_L\bar{\psi}_R) . \quad (1.1.11)$$

The first term allows the introduction of fermion masses, by setting  $g_f = \frac{m_f}{v}$ . The second term describes the interaction of fermions with the Higgs boson, the interaction strength of which is proportional to the mass of the fermion.

This mechanism had already been postulated in 1964. In 2012, a heavy scalar resonance with decay properties in line with predictions for the Higgs boson was discovered at the ATLAS and CMS experiments [15,16]. This discovery, in combination with measurements of the interactions and decay channels of the particle, widely confirmed the existence of the Higgs mechanism and presented a major step towards completing the standard model.

## 1.2 Physics Beyond the Standard Model

While the standard model accurately predicts a vast range of experimental observations and is itself a consistent theory, there are several phenomena not currently explained by the standard model. This hints at further physics beyond the standard model (BSM) that is yet to be discovered.

Several of these phenomena will briefly be covered in this section, however, this is by no means an exhaustive list.

### Dark Matter

There exist several pieces of evidence that suggest the majority of the known universe is not composed of visible matter, but rather so-called dark matter. One of the earliest indications was found in the rotational velocities of galaxies [17]. The majority of luminous matter tends to be concentrated in the central region of a galaxy. Therefore, the outer regions of a galaxy can largely be approximated as rotating around a central point mass. Under this assumption, one would expect the angular velocity of a given star to scale with the inverse square root of its distance from the galactic center. This is in conflict with actual observations which show an almost flat dependency between angular velocity and distance from the center, thereby implying the existence of additional, not visible mass in the galaxy, arranged in a halo around the center.

Further evidence of dark matter is given by the bullet cluster collision [18], where the apparent center of gravity of the collision does not match the center of gravity determined from the visible matter alone, indicating the existence of further, invisible matter. Additionally, structures found in the cosmic microwave background measured by the WMAP [19] and Planck [20] experiments are a strong indication that dark matter was present during the early formation of the universe.

There exists a wide range of models aiming to explain and describe dark matter, generally by introducing one or more new particles to the standard model. Several constraints can be placed on any dark matter candidate based on current observations; most importantly the lack of a dark matter discovery and seeming stability of dark matter put an upper limit on how strongly a dark matter particle can couple to the visible sector of the standard model.

Searches for dark matter candidates take several forms. In direct searches, one attempts to record interactions of dark matter particles with standard model particles using large, purpose-built detectors. In an indirect dark matter search, one looks for cosmic particles that could originate from the decay of dark matter into standard model particles.

Both these approaches largely aim to experimentally prove the existence of dark matter. However, probing the precise nature of a potential dark matter particle requires the controlled environment of a collider.

## Neutrino Mass

Within the current standard model, there are no right-handed neutrinos that would allow for a coupling between the Higgs and the neutrinos. Therefore, neutrinos are assumed to be massless particles. However, the observation of neutrino oscillation draws this assumption into question [21]. Neutrino oscillation describes a process, where the flavor of a neutrino switches to a different one as it propagates. One clear piece of evidence for this process is measurements of the flux of solar electron neutrinos. Here, the detected electron-neutrino rate does not line up with the rate that would be expected based on the fusion reaction occurring within the sun, indicating that some of the electron neutrinos oscillate into muon- or tau-neutrinos before they arrive at the earth. More recent measurements of both the total solar neutrino flux and solar electron-neutrino flux [69] confirm an abundance of non-electron-neutrinos originating from the sun.

This oscillation has several implications. For one, it indicates that neutrino flavor states are superpositions of several mass-eigenstates, therefore enabling the neutrinos to change from one flavor state to a different one. Additionally, the apparent time-dependent nature of the oscillation suggests that neutrinos do not propagate at light speed, as would be expected from a truly massless particle. Therefore, the existence of neutrino oscillation indicates that neutrinos have a small, but non-zero mass.

A popular theory to explain the neutrino masses is seesaw mechanisms [21,70] which introduce a heavy right-handed neutrino as a Majorana particle, meaning the right-handed neutrino would be its own antiparticle. In this model, the left-handed standard model neutrinos have a tiny chance to briefly convert into heavy right-handed neutrinos. This results in an overall small, but non-zero mass for the standard model neutrinos.

Beyond this, heavy right-handed neutrinos are also promising dark matter candidates. They can have a large enough mass to explain dark matter observations and have sufficiently small coupling to the standard model, as they are uncharged, color-neutral, right-handed particles and therefore do not interact directly via the strong, electromagnetic or weak interaction.

## Flavor Anomaly

In the standard model, the three lepton generations differ in their masses and flavor numbers but have an identical charge and iso-spin. Therefore, it is assumed that the coupling between gauge bosons and leptons is universal for all generations.

However, measurements performed by the LHCb collaboration [22,23] point towards a potential violation of the assumed lepton universality. These measurements investigated the decay of  $B^+$ -mesons, consisting of one anti-b-quark and one up-quark. One possible decay channel for  $B^+$ -mesons is  $B^+ \rightarrow K^+ l^+ l^-$ , where  $l$  is either a muon or electron. Assuming lepton universality, the rate of decay into electrons and decay into muons should only differ by a factor related to the mass difference between the two leptons. However, measurements of the relative rates showed a deviation of  $2.6\sigma$  from the expected ratio. While this is not sufficient for a discovery, it provides a hint at BSM physics.



Possible standard model extensions that could explain a violation of lepton universality are  $Z'$  models [71]. In such a model, the standard model is extended by an additional  $U(1)$  symmetry group that gives rise to a new neutral gauge boson, called a  $Z'$  due to its similarities to the standard model  $Z$ . Depending on the specific construction of the added  $U(1)$  symmetry, the resulting  $Z'$  may not couple to each lepton generation equally [72].

Beyond lepton universality,  $Z'$  models are of interest, as they offer a potential portal for interactions between the standard model and a dark matter candidate.

### 1.3 Collider Experiments

The likely existence of BSM physics provides a strong incentive to further probe the standard model. Efforts in pursuit of this can take various forms, such as searches directly looking for new, previously undiscovered particles, or measurements aiming to precisely determine standard model parameters in search of deviations from theoretical predictions. However, all these approaches require highly energetic particles in order to provide the necessary energy for the production of interesting heavy particles. In other words, they require powerful particle accelerators and colliders.

#### The Large Hadron Collider

The most powerful collider currently in operation is the Large Hadron Collider (LHC) [24] at CERN. The LHC is a circular proton-proton, Pb-Pb, and Pb-proton collider with a maximal center of mass energy of 14 TeV. The collider has been vital for particle physics ever since it went into operation, most notably for the discovery of the Higgs boson in 2012.

Modern particle colliders consist of two basic building blocks, radio-frequency cavities that accelerate a beam of charged particles and multi-pole magnets that focus or bend this beam. In a circular collider, or Synchrotron, such as the LHC, the magnets and cavities are set up to form a circular path. This allows the beam to repeatedly pass each cavity, receiving additional energy every time. However, when charged particles are deflected in a magnetic field, they radiate off energy in the form of Bremsstrahlung. In a circular collider, this energy loss is specifically known as synchrotron radiation. The exact amount of energy lost to radiation depends on the velocity of the particle, the radius of the path, and the mass of the particle. This places an upper limit on the energy a synchrotron can impart on a particle beam, as the particles will eventually be so fast that they radiate off as much energy as they receive from the accelerator. There are two ways to manipulate this limit. For one, larger bend-radius results in lower radiative losses, meaning larger colliders can reach higher energies. Further, heavier particles can be used to achieve higher energies in a synchrotron, as they have a lower velocity than a lighter particle with equal energy, and the cross section of Bremsstrahlung is smaller for particles with higher masses. For this reason, the LHC accelerates protons or lead ions, allowing it to reach significantly higher energies than its predecessor, the Large Electron–Positron Collider (LEP), despite using the same tunnel and therefore, having the same radius.

The LHC features two beam pipes, each containing a beam, circulating in opposite directions. Each beam is divided into 2808 bunches, where a bunch consists of approximately  $10^{11}$  particles. The beams are provided by a staged system of pre-accelerators and are injected into the LHC with an energy of 450 GeV. At four points around the accelerator, the beams are crossed and

brought to collision. Each interaction point is the site of one of the four large LHC experiments. Of these experiments, ATLAS [73] and CMS [74] are general-purpose experiments. LHCb [75] is specifically designed to investigate the physics of b-quarks, and ALICE [76] is an experiment specialized in heavy ion collisions.

Beyond the center-of-mass energy, a defining characteristic of a collider is the rate at which particle interactions occur, described by the instantaneous luminosity  $\mathcal{L}$  of the collider. It can be calculated using the expression

$$\mathcal{L} = \frac{N_b^2 k_b f}{A} H_D, \quad (1.3.1)$$

where  $N_b$  is the number of particles per bunch,  $k_b$  is the total number of bunches in the collider and  $f$  is the bunch crossing rate. The quantity  $A$  describes the size of the area in which the bunches are crossed, and  $H_D$  is a correction term accounting for the widening of the bunches due to their own electric field. The original design luminosity of the LHC was  $\mathcal{L} = 10^{34} \text{cm}^{-2} \text{s}^{-1}$ , however the currently ongoing high luminosity LHC upgrade [25] aims to increase this by approximately a factor of 5 through reduction of the beam crossing area.

Equally relevant as the instantaneous luminosity is the integrated luminosity  $\mathcal{L}_{\text{int}}$ , defined as

$$\mathcal{L}_{\text{int}} = \int \mathcal{L}(t) dt, \quad (1.3.2)$$

with the integral running over a given interval of time. Using the integrated luminosity one can calculate the expected amount of occurrences of a specific process with a cross section of  $\sigma_{\text{proc}}$  as

$$N_{\text{proc}} = \mathcal{L}_{\text{int}} \sigma_{\text{proc}}. \quad (1.3.3)$$

## Detectors

In order to detect particles produced in high-energy collisions, modern experiments use highly sophisticated detector setups. Contemporary general-purpose detectors are subdivided into multiple sub-detector systems, arranged similarly to the layers of an onion. Each sub-detector is specialized to measure the properties of a set of particles. Figure 1.2 shows an example of such a multi-layers setup using the CMS detector. The individual sub-detectors will briefly be covered in the following.

The *tracking system* or *tracker* generally directly surrounds the interaction points. It is designed to capture the paths of charged particles while providing as little interfering material as possible for the particles to interact with.

The detector itself is suffused with a strong magnetic field  $\mathbf{B}$ , with field lines running parallel to the beam pipe. Charged particles traversing this magnetic field with a velocity vector  $\mathbf{v}$  experience the Lorentz force along the direction of  $\mathbf{v} \times \mathbf{B}$ . This causes the paths of charged particles to be bent around the beam direction. The radius  $R$  of the bend is given by

$$p \cos \lambda = 0.3BR, \quad (1.3.4)$$

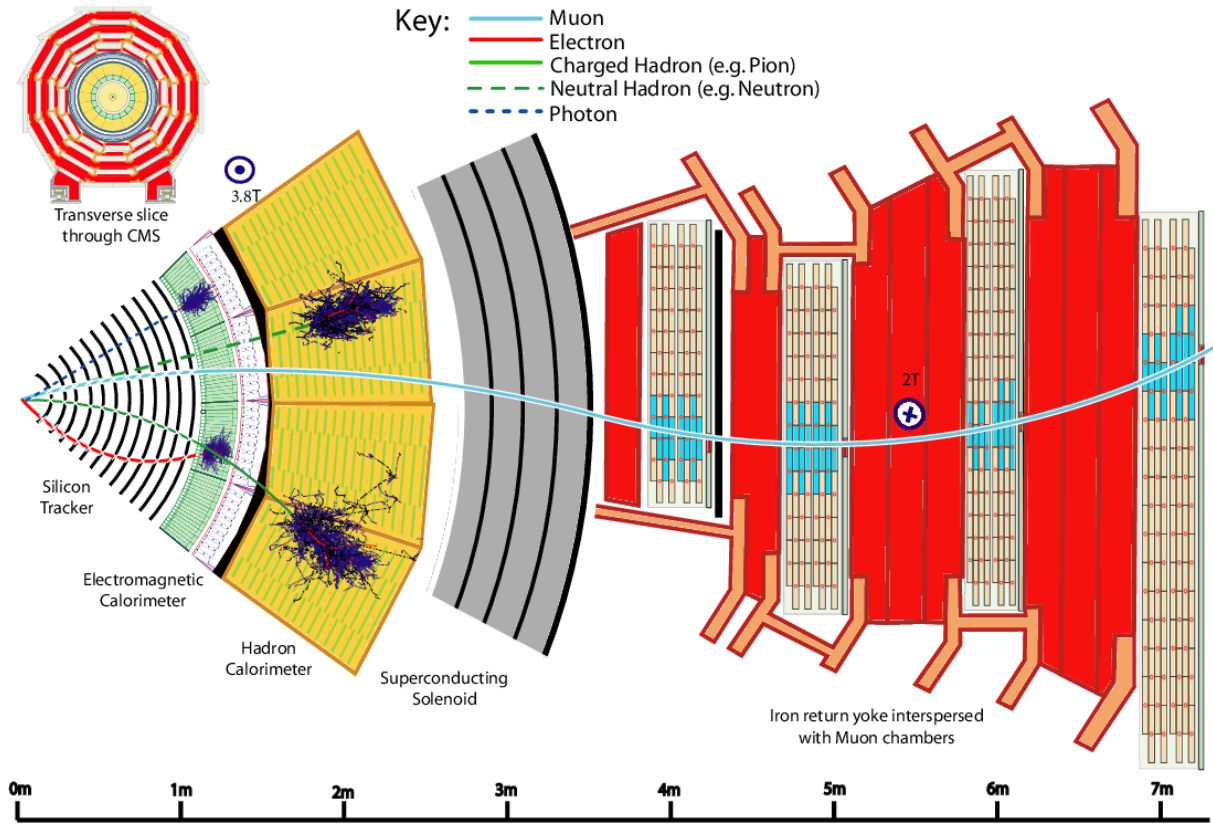


Figure 1.2: Schematic view of a slice of a general particle detector, using the CMS detector as an example. Figure taken from [77].

where  $B$  is the strength of the magnetic field,  $\lambda$  is the pitch angle between the path of the particle and the plane orthogonal to the beam direction, and  $p$  is the momentum of the particle [66]. This allows for the reconstruction of the momentum of a charged particle, based on its path, recorded by the tracker. However, for high momentum particles, the bend radius can become very large, resulting in a only slightly bent track. Therefore the momentum resolution of a tracker is lower for highly energetic particles.

There exists a multitude of tracker designs. One approach is silicon trackers, such as the one used in CMS. These systems consist of several concentric layers of silicon pixel and strip detectors. When a charged particle traverses the silicon, it will create electron-hole pairs in the silicon that can be detected. From all these individual hits, the track of the particle can then be reconstructed. Another approach is time projection chambers (TPCs). A TPC consists of a large, cylindrical, gas-filled chamber. As charged particles traverse the chamber they ionize gas atoms along their path. The electrons freed in the ionizations are drawn by a combined electric- and magnetic field towards one of the flat ends of the TPC. There the positions of the electrons in the  $x, y$  plane are recorded. Additionally, the time between the interaction and the detection of the electrons can be used to reconstruct the  $z$  position of the ionization. Combining both of these measurements results in a full description of the path of the traversing particle.

Only electromagnetically charged particles can cause these ionizations in the tracking system, meaning that a tracker cannot be used to record the paths of electromagnetically uncharged

particles. Therefore, the properties of electromagnetically neutral particles have to be measured by different means.

Surrounding the tracker is the *calorimeter* system. These detectors provide large, dense volumes of material for particles to deposit their energy in. The deposited energy can then be read out and used to determine the energy of the original particle.

Both neutral and charged particles can deposit their energy in a calorimeter. The energy resolution of the tracker tends to be better than the resolution of a calorimeter for charged particles, however, calorimeters present the only way to detect and measure neutral particles in a collision event. More details on the operation principles and construction methods of calorimeters are covered in Chapter 2.

The outermost sub-detector is the *muon system*. Due to their high mass and limited interactions, muons only deposit a small fraction of their energy in the calorimeter and therefore continue to propagate beyond the calorimeter system. The muon system operates on a similar principle to the tracker, in that it aims to record the paths of these muons. However, the spatial resolution of the muon system tends to be lower than that of the tracker, as it has to cover a significantly larger volume.

The muon system serves two purposes. For one, it allows for the identification of muons, as any single, highly energetic particle that leaves the calorimeter is likely to be a muon. Beyond this, the paths recorded by the muon system provide a second point of measurement for the track bend of a muon, thereby increasing the precision with which muon energies can be determined.

## 1.4 Linear Lepton Colliders

Hadron colliders like the LHC are capable of providing high center-of-mass energies due to the high mass of the protons they accelerate. However, the composite nature of the proton can present a problem. A proton mainly comprises two up quarks and one down quark. In addition to these so-called valence quarks several further quark-antiquark pairs, known as sea quarks can be present, as well as gluons mediating the QCD interactions between the quarks.

The total energy of the proton is not bound in any one of its constituents but probabilistically distributed, according to a parton density function (PDF) [78], between all of them. Figure 1.3 shows proton PDFs for two squared energy scales  $Q^2$ . The lines give the probabilities, normalized to the total amount of particles in the proton, of finding a specific particle with a fraction  $X$  of the total proton energy.

During a proton-proton collision, the actual interactions take place between the constituent particles. This means the energy available in the interaction is only a fraction of the total proton energy, and the exact value of this energy is probabilistically distributed, rather than deterministic. This makes precision measurements at a hadron collider difficult.

Using leptons instead of protons would allow for precise tuning of the energy available in each interaction, as leptons are not composite particles. However circular electron colliders run into the aforementioned problem of losing a significant fraction of their energy to synchrotron radiation, and the short lifetime of the muon makes the prospect of a muon collider difficult to implement in practice. Therefore, there have been recent proposals for linear lepton colliders, which are not limited by synchrotron radiation losses, and still provide the clean collision environment of a lepton collider.

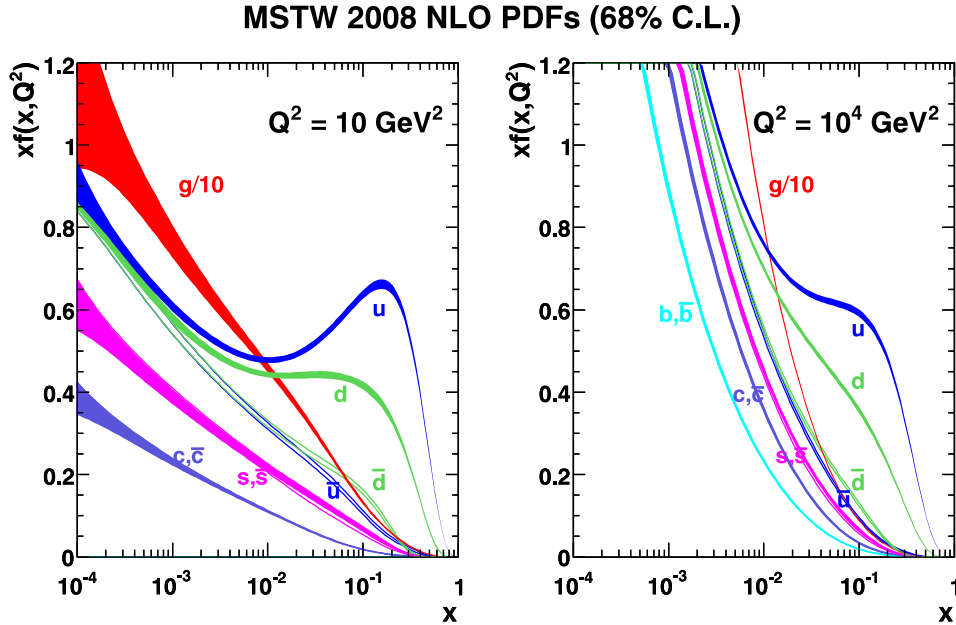


Figure 1.3: PDFs for a proton from the MSTW NLO PDF set [78]. PDFs are shown for  $Q^2 = 10\text{GeV}^2$  (left) and  $Q^2 = 10^4\text{GeV}^2$  (right). The filled region corresponds to the 68% confidence interval. Figure taken from [78].

## International Linear Collider

The International Linear Collider (ILC) [26] is one such proposed linear collider. The ILC is designed to accelerate polarized electron and positron beams in two 11km long linear accelerators using superconducting RF cavities. The maximal center-of-mass energy of the design is 500 GeV, although a staged approach with an initial energy of 250 GeV is under consideration. The individual components and the proposed layout of the ILC are shown in Figure 1.4. Combined limitations of the electron and positron sources, the RF cavities, and the cryogenic system [26] make it infeasible for the ILC to accelerate a constant beam. Instead the linear accelerators operate in bunch trains. Every 200ms a group of 1312 bunches is accelerated over the course of approximately 1ms, followed by a downtime of about 199ms until the next group of bunches is accelerated.

One of the goals of the ILC is to use its precise control over the center of mass energy to perform precision measurements of the Higgs boson and its couplings to other particles. Further, the use of polarized beams allows for the suppression or enhancement of specific processes, potentially reducing certain backgrounds.

The ILC plans to employ a dual detector setup, where two detectors can be alternately placed at the interaction point. The two proposed detectors are the International Large Detector (ILD) [59] and the Silicon Detector (SiD) [79]. Both detectors aim to reconstruct every particle in a collision event using the particle flow approach [80]. This necessitates high tracking and calorimeter resolution. The ILD aims to realize this with a tracking system that combines a silicon tracker directly surrounding the beamline with a TPC, as well as with a highly granular

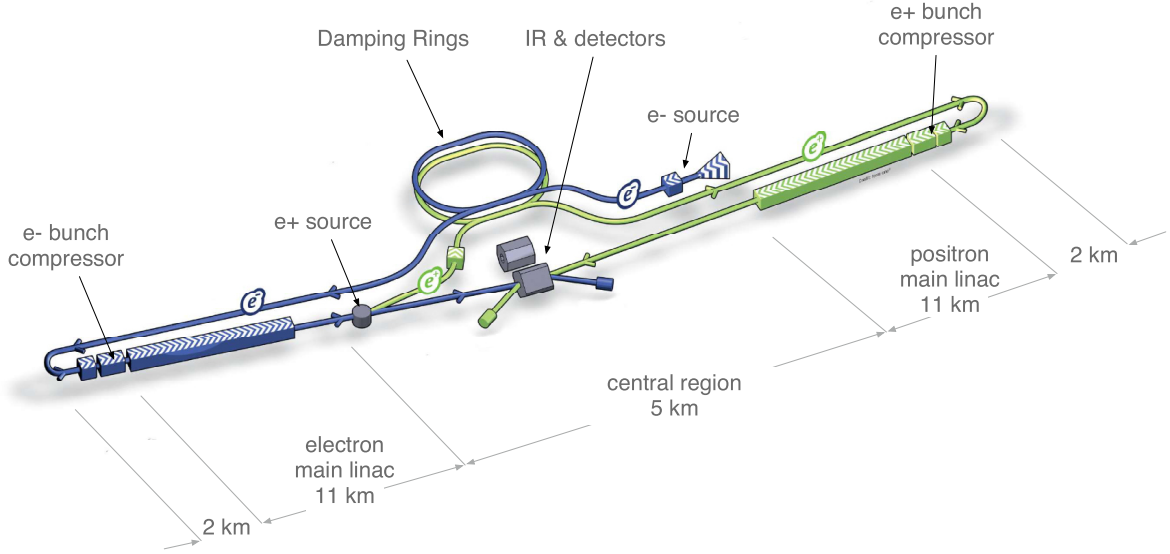


Figure 1.4: Schematic drawing of the ILC components and their layout, component sizes not to scale. Figure taken from [26].

calorimeter setup. The SiD plans to use an entirely silicon-based tracker, in addition to a calorimeter system comparable to the ILD. The pulsed nature of the ILC beam offers several design possibilities for the detectors, such as the use of power-pulsing, where the detectors are powered off during the downtimes between bunch trains, which significantly reduces the waste heat produced by the detector, thereby eliminating the need for active cooling.

### Compact Linear Collider

The Compact Linear Collider (CLIC) [27, 81] presents another proposal for a linear electron-positron collider to be constructed at CERN.

The CLIC concept employs a novel two-beam setup, where a secondary, high-current drive beam runs in parallel to the main linear accelerators. The accelerator systems then extract energy from this drive beam to power the RF cavities accelerating the main electron and positron beams. A depiction of this layout can be seen in Figure 1.5. The initial target center-of-mass energy of the collider is 380 GeV, however, the two-beam setup has been specifically designed to allow for staged upgrades to center-of-mass energies of 1.5 TeV and 3 TeV.

## 1.5 Precision Measurements at Lepton Colliders

The current standard model is a self-consistent theory, however several parameters, most notably particle masses and coupling constants of interactions, are not directly derived from theory and have to be experimentally measured. The lack of discoveries of new particles indicates that these particles either exist at energy scales not currently accessible or that their effect on observations is too subtle to be easily detected. If the latter is the case, it is vital to have a precise understanding of the standard model, in order to spot even slight deviations between theoretical predictions and

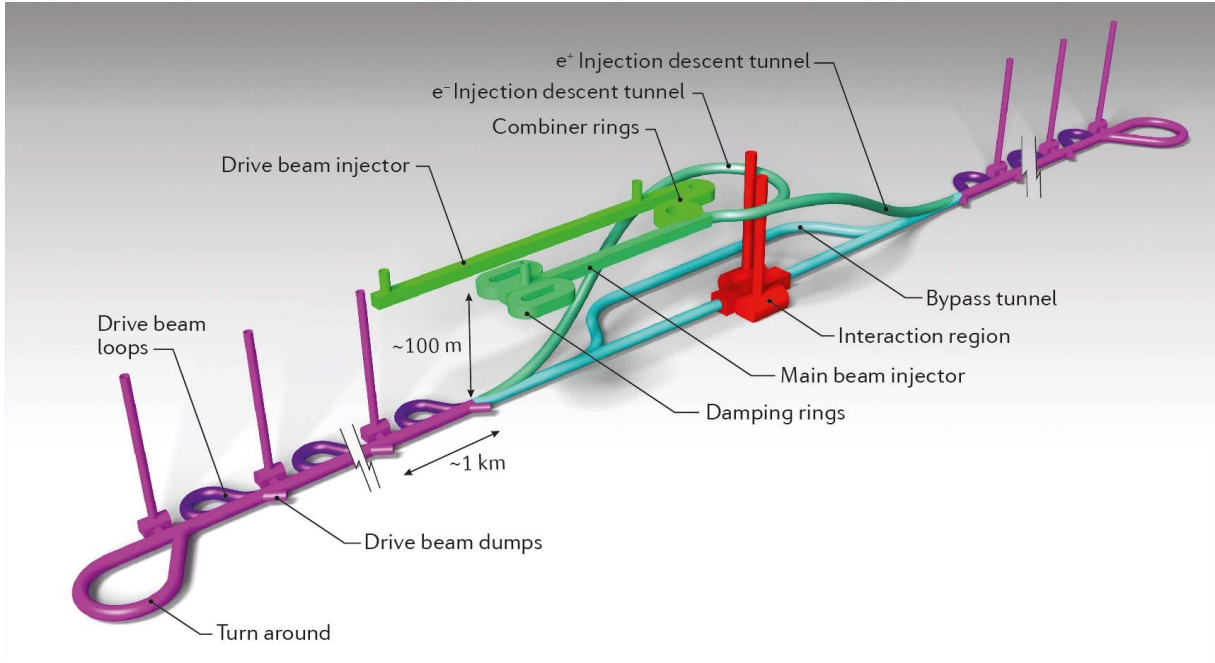


Figure 1.5: Schematic drawing of the CLIC layout. Figure taken from [82].

measured data. This makes exact measurements of standard model parameters an important effort.

The wide range of collision energies and large QCD-induced backgrounds make precision measurements at a hadron collider difficult, however, a lepton collider, in combination with a detector capable of reconstructing every particle in a collision event, would enable these measurements.

## Higgs Measurements

The Higgs resonance discovered at the LHC offers a clean way of explaining the mass of bosons and fermions, however, the exact properties of the discovered boson have yet to be measured in detail.

Noteworthy among the properties of the Higgs boson are the branching ratios of its decay channels. For a general particle  $A$ , its branching ratio  $\text{BR}_{A \rightarrow XY}$  describes the probability for the particle to decay into a specific set of decay products  $X$  and  $Y$  [66]. The exact value can be determined using the expression

$$\text{BR}_{A \rightarrow XY} = \frac{\Gamma_{A \rightarrow XY}}{\Gamma_A}, \quad (1.5.1)$$

where  $\Gamma_{A \rightarrow XY}$  is the partial decay width for the specific process and  $\Gamma_A$  is the total decay width of the particle defined as the sum over the partial width of all possible decay channels. The partial width  $\Gamma_{A \rightarrow XY}$  is proportional to the squared coupling of  $A$  to  $X$  and  $Y$ .

For the Higgs boson, its direct couplings to other particles are determined by the masses of the particles and by the vacuum expectation value, which is known from the Higgs mass.

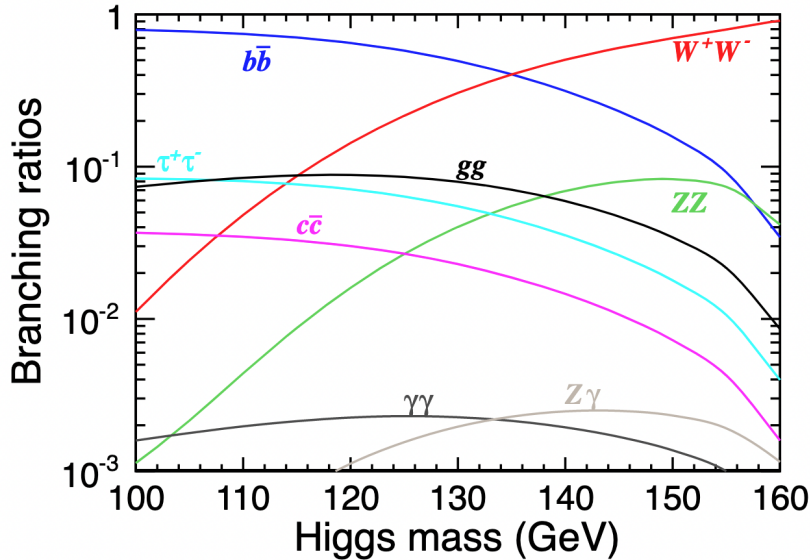


Figure 1.6: Theoretical predictions of the Higgs branching ratios, for a Higgs boson with a mass between 100 GeV and 160 GeV. Figure taken from [83].

Therefore, the branching ratios of the Higgs boson can be precisely predicted. Figure 1.6 shows these predictions for a range of Higgs boson masses. This makes direct measurements of Higgs decays a promising way to probe the physics of the Higgs sector.

The main decay channel of a 125 GeV Higgs boson is  $H \rightarrow b\bar{b}$ . However, this channel is exceedingly difficult to measure at a hadron collider, as the large QCD background results in an abundance of  $b$ -quark pairs that need to be separated from those produced by Higgs decays.

In a lepton collider setting the lack of QCD-induced backgrounds allows for significantly more precise measurement and selection of  $H \rightarrow b\bar{b}$  decays.

Further, a lepton collider allows for direct measurements of the Higgs boson mass and decay width without having to rely on observation of its decay products. The most likely Higgs production process in an electron-positron collision with a center of mass energy of 250 GeV is Higgs Strahlung [83], where a Higgs boson is produced in association with a  $Z$  boson. The  $Z$  has a chance to continue on to decay into an electron or muon pair, the momentum of which can be measured to high precision using the tracking and muon systems. From this, the original  $Z$  momentum can be reconstructed. In a hadron collider, the exact center-of-mass energy of the interacting partons is random. Additionally, the two interaction partners do not necessarily have the same momentum, meaning the center of mass of the collision will likely have momentum along the beam direction. In a lepton collision, however, the center-of-mass energy is precisely known, and the collision happens effectively at rest. Therefore, the momentum and energy of a  $Z$  produced through Higgs Strahlung in a  $e^+e^-$  collision can be compared to the known collision parameters in order to determine the energy and momentum of the produced Higgs. This can, in turn, be used to measure the mass and width of the Higgs boson to high precision. Most notably, this measurement can be used to detect Higgs events, even if the Higgs boson itself decays into an invisible final state, making the measurement a promising avenue for dark matter searches.



## Precision Electroweak Measurements

$Z'$  models present a popular set of standard model extensions, potentially offering explanations for flavor anomalies and providing a mediator to dark matter. However, searches at the LHC have shown no direct observations of a heavy  $Z'$  up to a mass of 3 TeV. Moving to even higher collision energies presents a daunting challenge, however, there are indirect measurements that can be used to search for the existence of a  $Z'$ , even at lower energies.

If a heavy  $Z'$  that couples to fermions  $f$  exists, then it may affect the rate of  $e^+e^- \rightarrow f\bar{f}$  processes, even at energies too low to produce the  $Z'$  on-shell [83]. Therefore, measurements of the  $e^+e^- \rightarrow f\bar{f}$  provide a chance to discover BSM physics, however, due to the likely small effect of a potential  $Z'$ , any such measurement requires high precision.

The large QCD background at the LHC makes such precision measurements difficult. On the other hand, the measurements performed at LEP demonstrate that a lepton collider is a particularly well-suited environment for achieving this needed level of precision.

## 1.6 Monte Carlo Simulation

When particles interact, there exists a multitude of potential processes that can occur. Each process has a specific likelihood, however, which process actually happens is random. For a process with a cross  $\sigma_{\text{proc}}$  and for an integrated luminosity  $\mathcal{L}_{\text{int}}$ , the expected number of occurrences can be determined using Equation 1.3.3. However, in practice, not every instance of a process is detected. The expected number of observed processes is therefore given by

$$\bar{N}_{\text{obs}} = \mathcal{L}_{\text{int}}\sigma_{\text{proc}}\eta_{\text{det}} , \quad (1.6.1)$$

Where  $\eta_{\text{det}}$  is the detection efficiency for this process. The actual number  $n$  of detected interactions follows Poissonian statistics and is described by the distribution

$$p(n) = \frac{\bar{N}_{\text{obs}}^n}{n!} e^{-\bar{N}_{\text{obs}}} . \quad (1.6.2)$$

The variance of this distribution is given by  $\sqrt{n}$ , and the relative uncertainty of an observation of  $n$  points can be written as

$$\sigma_{\text{rel}} = \frac{n}{\sqrt{n}} . \quad (1.6.3)$$

Therefore in order to reach sufficient precision for the measurements outlined in Section 1.5, one needs to increase the number of observed points. This can be achieved by improving the detection efficiency, however, the efficiency is per definition constrained to the range between 0 and 1. To achieve precision beyond this, an increase in luminosity is required. For this reason, modern collider experiments move to higher and higher luminosities.

An important principle of the scientific method is to define a theory, use this theory to make a prediction and finally compare the prediction to experimental observation.

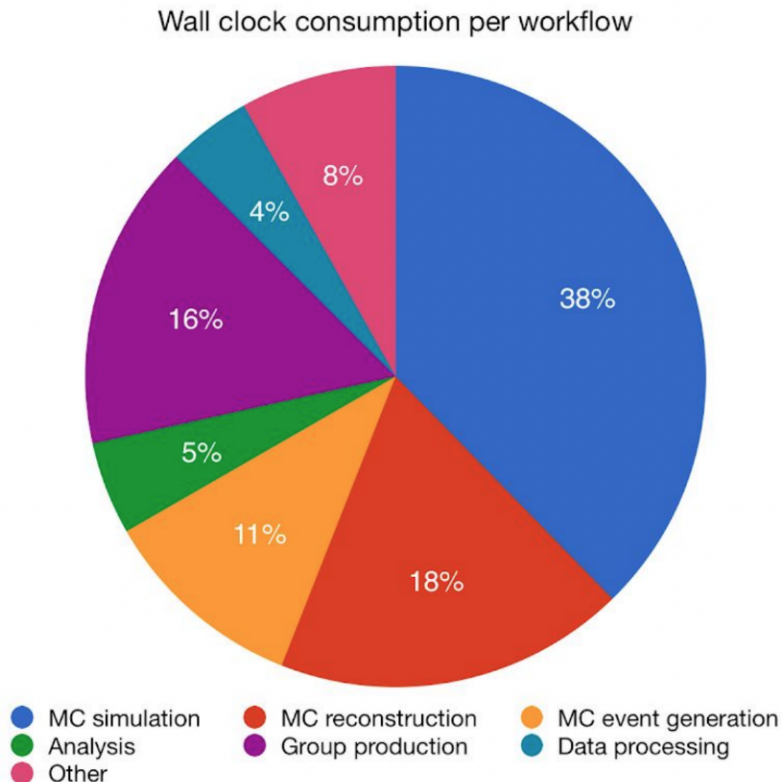


Figure 1.7: Fractions of compute times that were used for various activities by the ATLAS collaboration in 2018. Figure taken from [30].

In particle physics, the step from theory to prediction consists of modifying the standard model Lagrangian according to the new theory and then modeling how these changes manifest as observations in the experimental measurement. It is currently not feasible to directly jump from Lagrangian to measurement. Therefore the simulation process is split into several parts that make up the full simulation chain.

The first step in the simulation chain is *event generation*. Here, the hard scattering processes between the initial state beam particles are modeled. This requires calculating the cross section of individual possible processes. To this end, the matrix element of each process has to be determined, potentially including higher-order corrections. Then the differential cross section has to be integrated over a high dimensional phase space in order to obtain the total cross section while observing momentum conservation and other constraints. This makes the analytic calculation of the phase space integrals not feasible. Instead, event generators make use of numerical integration techniques, most commonly MC integration.

The end result of the event generation consists of a list of particles produced in the initial interaction, where each particle is described by its particle type and its momentum.

Among the particles produced by the event generation are individual quarks and gluons. However, these particles cannot propagate freely due to color confinement and instead form a jet, a collimated spray of color-neutral particles. The simulation of this process is known as

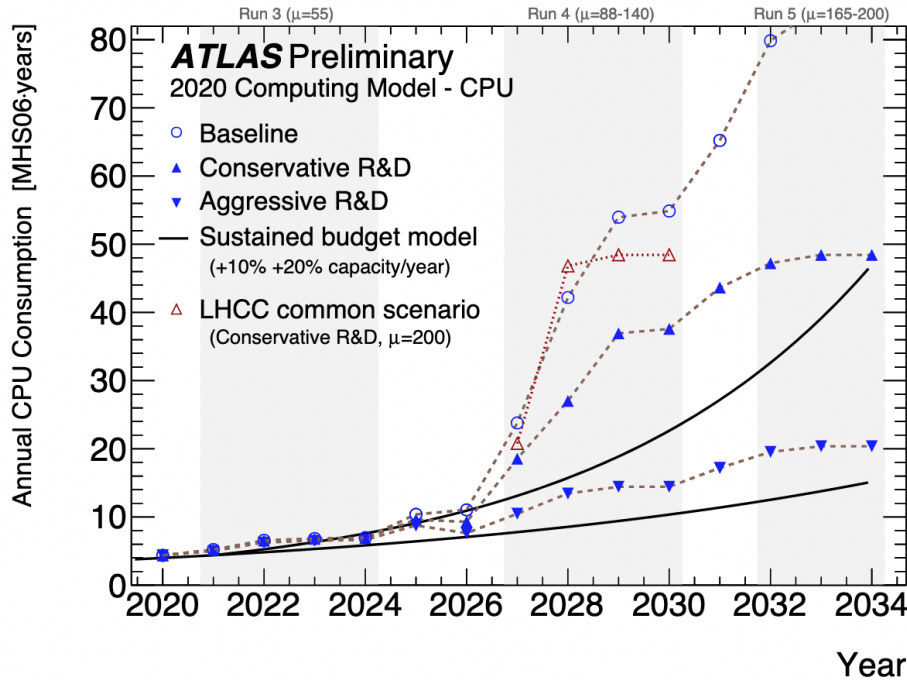


Figure 1.8: Predicted computational resource requirement in MHS06 [84] estimated by the ATLAS collaboration for various R&D scenarios described in Reference [30]. Figure taken from [30].

*hadronization*. Additionally, the decays of particles with lifetimes short enough to decay within the detector are modeled in this step.

The final step of the simulation chain consists of *detector simulation*. This step takes the particles produced by event generation and hadronization and models their interaction with the individual detector components, as well as modeling the detector signals these interactions result in. A large number of particles that reach the detector can make this simulation a time-intensive process.

The end result of this simulation chain are simulated events, often referred to as MC data, which can be compared to direct experimental observation. This comparison can then be used to test new particle physics theories. However, the accuracy of this comparison is determined by both the statistical uncertainty of the measured data and of the MC data. These uncertainties follow Equation (1.6.3) and thereby scale with the amount of available data. In order to not be limited by the MC data uncertainty, one, therefore, aims to have a number of simulated samples that is at least equal to the number of measured events. This leads to an escalating need for MC data, as colliders move to higher luminosities. This presents a problem, as MC data generation already occupies a significant portion of the high energy physics (HEP) computing budget. For example, the ATLAS collaboration dedicated around half of its computational capacities in the year 2018 to event generation and detector simulation [30], as shown in Figure 1.7.

Therefore, the escalating need for MC data directly translates to an escalating need for computational resources. Estimates by the ATLAS collaboration, shown in Figure 1.8, predict

that this requirement will exceed the available computing budget by 2028, unless aggressive R&D measures are taken.

This makes the development of faster, more efficient simulation methods vital for future precision measurements. One such method, generative ML models, is the main focus of this work.

## Chapter 2

# Calorimetry

Modern collider experiments aim to probe fundamental interactions between elementary particles at unprecedented precision. To this end, two beams of highly energetic particles are brought to collision and the processes that occur during these collisions are investigated. While the interactions themselves cannot be directly observed, their end results, so-called final-state particles, can be measured.

To perform these measurements, several types of sub-detectors have been developed to record specific features of final state particles. Some of the most vital observables to capture are the energies of final state particles. For charged particles, for example electrons and muons, this energy can be reconstructed based on their bent path through the magnetic field of the detector. However, for neutral particles like photons or neutral pions, this is not an option, meaning that their energies have to be measured directly through calorimeters.

Fundamentally, a calorimeter is a larger volume of material, in which a particle deposits its energy. This deposition takes the form of particle showers, massive cascades of secondary particles that all deposit energy within the calorimeter. One can differentiate between two types of showers, electromagnetic showers, and hadronic showers. As the names imply, electromagnetic showers are caused by stable particles that only interact via electromagnetic interaction, such as electrons and photons. Conversely, a hadronic shower originates from the interaction of a stable hadron, such as a proton, neutron, pion, or kaon, with the detector material.

In this chapter, we will initially cover the interactions and processes that contribute to the formation of electromagnetic showers in Section 2.1. Following this, Section 2.2 describes the interactions heavy charged particles undergo in a calorimeter. Section 2.3 details the properties of hadronic calorimeter showers. Current approaches for the simulation of calorimeter showers are discussed in Section 2.5 and finally, we introduced the calorimeter systems of the ILD in Section 2.6.

### 2.1 Electromagnetic Showers

There are several fundamental processes in which photons and electrons interact with matter. The likelihood for a certain process to occur depends heavily on the energy of the particle.

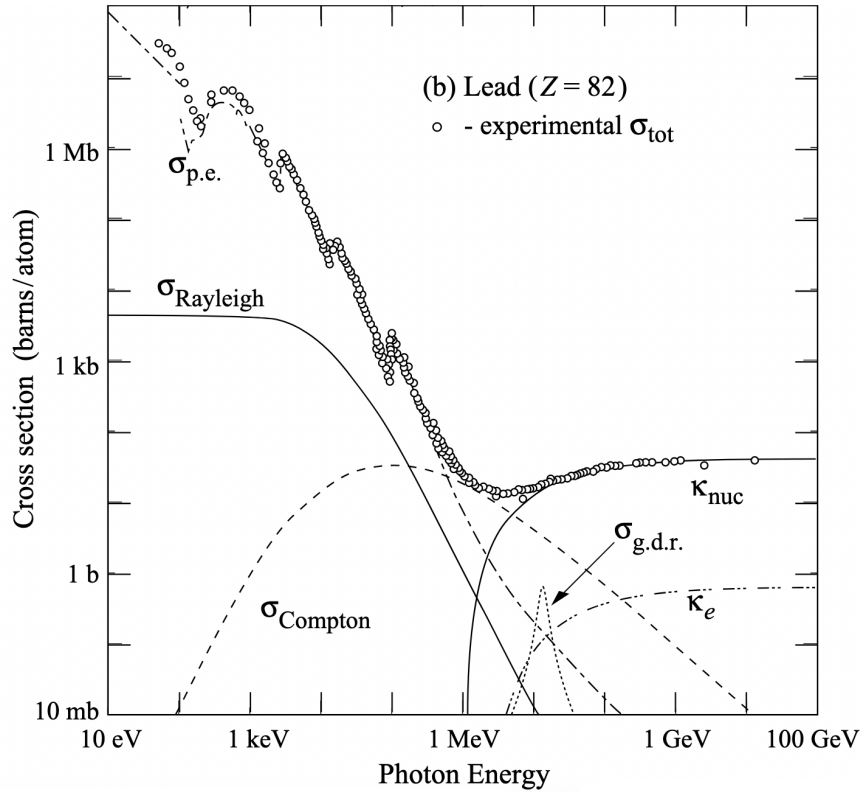


Figure 2.1: Cross sections of various photon-matter interactions. The abbreviation p.e. corresponds to the photoelectric effect, and  $\kappa_{\text{nuc}}$  and  $\kappa_e$  describe the cross section of pair production in the electric field of a nucleus and in the electric field of an electron respectively. Figure taken from [85].

## Photon-Matter Interactions

Photons mainly interact with the atoms of material via the *photoelectric effect*, *Rayleigh scattering*, *Compton scattering*, or *pair production*. Figure 2.1 shows the cross sections of these individual processes. Their details will be discussed in the following.

In the *photoelectric effect*, the photon is absorbed by one of the electrons of the atom and the absorbed energy causes the electron to be ejected from the atom. The cross section of the photoelectric effect drops with increasing photon energy, causing the photoelectric effect to be the most probable process for lower energetic photons. It remains dominant up until around 1 MeV, where it is overtaken by Compton scattering.

*Rayleigh scattering* and *Compton scattering* both describe scattering processes of a photon with an electron. In Rayleigh scattering, this process is an elastic interaction with a bound electron. This produces a scattered photon with the same energy as the original photon. The cross section of Rayleigh scattering drops off for large energies, similar to the cross section of

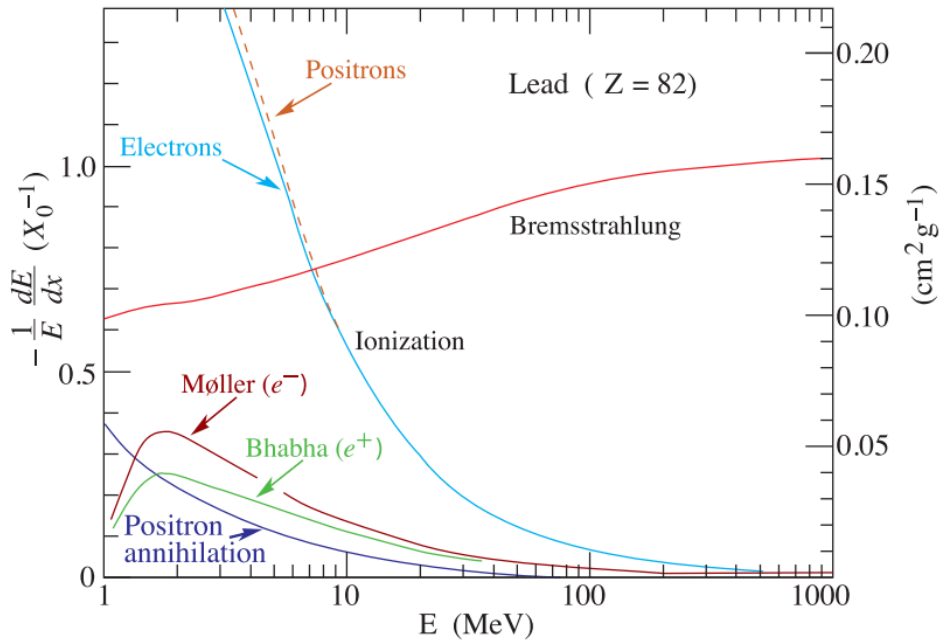


Figure 2.2: Energy loss per radiation length in lead for various electron-matter interactions. Taken from [86].

the photoelectric effect. Compton scattering, on the other hand, is the inelastic interaction with a free electron in which the scattered photon loses some of its energy to the electron. In order for Compton scattering to occur with a bound electron, the photon energy needs to be high enough that the binding energy of the electron becomes negligible. Therefore, the cross section of Compton scattering is suppressed for low energies, and the process only becomes dominant in the region between 1 MeV and 10 MeV. Beyond 10 MeV it is superseded by pair production.

In *pair production*, a photon interacts with the electromagnetic field of an electron or nucleus to form an electron-positron pair. Pair production can only occur if the photon energy is at least equal to the rest mass of the electron-positron pair of  $2 \times 511 \text{ keV} = 1.022 \text{ MeV}$ . At energies of 10 MeV pair production is the most probable process and it remains the dominant process for energies beyond this.

## Electron-Matter Interactions

Similar to photons, the interaction of electrons/positrons with matter can be categorized into a set of processes with energy-dependent cross sections. Figure 2.2 shows the cross sections of these processes. There are some differences in the exact behavior between electrons and positrons, as matter contains electrons, but no positrons; however, the overall processes are still similar.

*Ionization* in this case describes an interaction between the energetic electron and a bound electron, in which part of the energy is transferred to the bound electron. This causes the bound electron to effectively be knocked loose, leaving behind an ionized atom. Ionization is by far the dominant process in the low energetic region up to 1 MeV.

*Bhabha scattering* and *Møller scattering* are the scatterings of a positron or electron with an electron of the material, respectively. Overall, both scattering processes have relatively small cross sections compared to both ionization and Bremsstrahlung.

*Bremsstrahlung* is a process in which an energetic electron interacts with the electromagnetic field of a nucleus and thereby radiates off a part of its energy. This results in a less energetic electron and a photon. For energies above 10 MeV Bremsstrahlung becomes the most likely interaction between electrons and matter.

## Showers

One can see that highly energetic photons interacting with matter tend to lose their energy by creating electrons through pair productions, and highly energetic electrons and positrons tend to create photons through Bremsstrahlung. This means that if, for example, a photon enters material, it can undergo pair production. The produced electron and positron will themselves interact with the material, most likely via Bremsstrahlung, resulting in a photon and electron (positron) each. The photons can then undergo further pair production, while the electrons and positrons continue to radiate off more photons via Bremsstrahlung and so on and so forth. The end result is a cascade of particles that is called a particle shower. Figure 2.3 shows a rendition of such a cascade caused by repeated Bremsstrahlung and pair production. This cascade continues to develop until the average electron energy drops below the critical energy  $E_c$ , where ionization and Bremsstrahlung have approximately the same cross section. Below this threshold particles tend to deposit their remaining energy in the material without further splitting. Both showers induced by photons and by electrons mainly consist of the same types of particles undergoing the same types of interactions. This causes electron and photon showers to be similar in appearance.

When building a calorimeter it is important to know the average size of a shower, so it can be ensured the shower is fully contained within the detector. The characteristic length scale of a photon shower is defined by its *radiation length*  $X_0$ , which describes the average length between Bremsstrahlung interactions. This length is approximated by the expression [88]

$$X_0 = \frac{716.4A}{Z(Z+1)\ln\left(\frac{237}{\sqrt{Z}}\right)} \frac{\text{g}}{\text{cm}^2}, \quad (2.1.1)$$

where  $Z$  is the atomic number and  $A$  is the atomic mass number. However, an EM shower will consist of a multitude of interactions that all need to be captured. Therefore, an EM calorimeter typically has a depth of 15-30  $X_0$  [85]. Every pair production and Bremsstrahlung process results in a pair of particles with approximately half the energy of the original particle, and the overall depth of an electromagnetic shower scales with the logarithm of the energy of the initiating particle.

The transversal extent of an electromagnetic shower is described by the Moliere radius  $R_m$ , defined by the radius around the impact of the initiating particle, which, on average, contains 90% of the shower energy. The Moliere radius can be approximated as [85]

$$R_m = X_0 \frac{E_s}{E_c}, \quad (2.1.2)$$

where  $E_c$  is the critical energy and  $E_s$  is the scale energy defined as



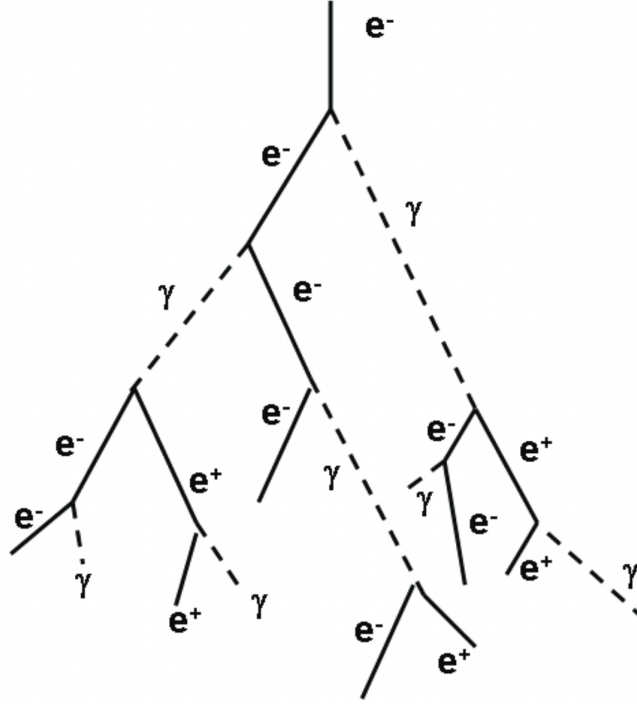


Figure 2.3: Schematic depiction of an electromagnetic shower. Figure taken from [87].

$$E_s = \sqrt{\frac{4\pi}{\alpha} m_e c^2} \approx 21.2 \text{ MeV}, \quad (2.1.3)$$

with the electron mass  $m_e$  and the fine-structure constant  $\alpha$ .

## 2.2 Heavy Charged Particles

Heavy charged particles can undergo the same Ionization, Bhabha scattering, Møller scattering, and Bremsstrahlung processes as electrons. However, the Bremsstrahlung cross section scales with  $\frac{1}{m^4}$ , where  $m$  is the mass of the particle traversing the material. Therefore, Bremsstrahlung is no longer the dominant process for heavy charged particles, and highly energetic heavy charged particles mainly lose energy through ionization instead. The average energy loss per distance is described by the Bethe-Bloch equation [89],

$$\left\langle -\frac{dE}{dx} \right\rangle = K z^2 \frac{Z}{A} \frac{1}{\beta^2} \left[ \frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 W_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right], \quad (2.2.1)$$

where  $A$  and  $Z$  are the atomic weight and number of the material,  $z$  is the charge of the particle and  $K$  is a numerical constant.  $m_e$  is the electron mass,  $\beta$  is the velocity, and  $\gamma$  is the Lorentz factor of the ionizing particle. Further,  $I$  describes the excitation of the material, and  $W_{\max}$  is the maximal kinetic energy transferred in an interaction. Finally,  $\delta(\beta\gamma)$  is a correction factor for

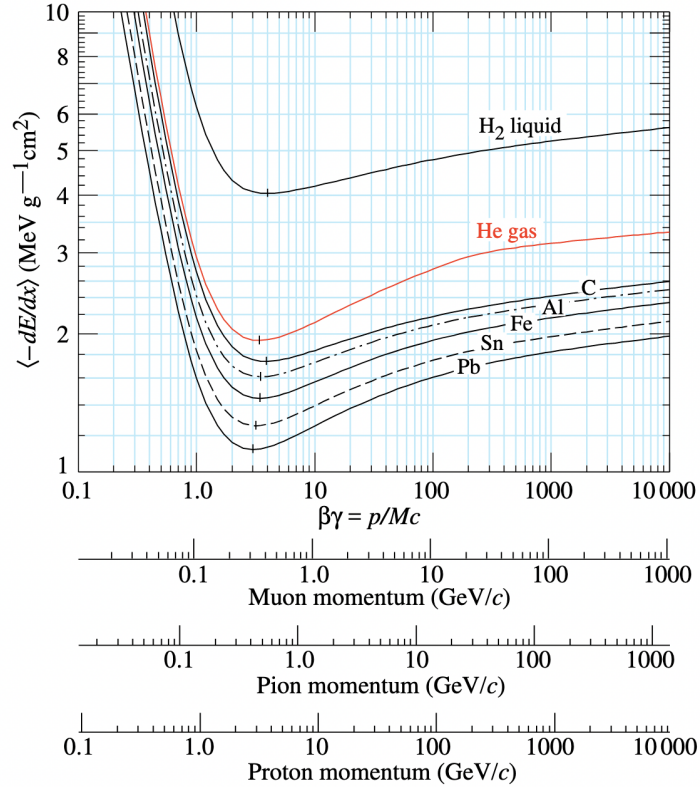


Figure 2.4: Energy loss through ionization in various materials for muons, pions, and protons, as described by the Bethe-Bloch equation. Figure taken from [89].

density effects, that depends on the momentum of the particle. Figure 2.4 shows energy losses for a range of materials as a function of the momentum of the particles.

From Figure 2.4 we can also see that the energy loss through ionization reaches a minimum around a particle momentum of  $\beta\gamma \approx 2 - 3$ , with only a small increase in the energy loss when moving to higher momenta. A particle with energy in the vicinity of this minimum is known as a minimum ionizing particle (MIP) and loses only a small amount of energy per ionization. This leads to the particle energy remaining nearly unchanged throughout many interactions, and all of these interactions result in energy deposits of similar magnitude. The energy deposition of a minimum ionizing particle is commonly also referred to as a MIP. As the MIP depositions are largely independent of the precise energy of the particle, they play an important role in the calibration of calorimeters. Once the MIP loses enough energy to drop below a momentum of  $\beta\gamma \approx 1$ , the average ionization energy loss increases and the particle quickly deposits its remaining energy. While only muons are heavy, stable, and non-interactive enough to traverse material while only losing minimal energy through ionization, every charged particle with momentum around the minimum of the Bethe-Bloch equation can cause multiple MIP energy depositions. Within a particle shower, this means that MIP depositions occur more often than other ionization energy losses.

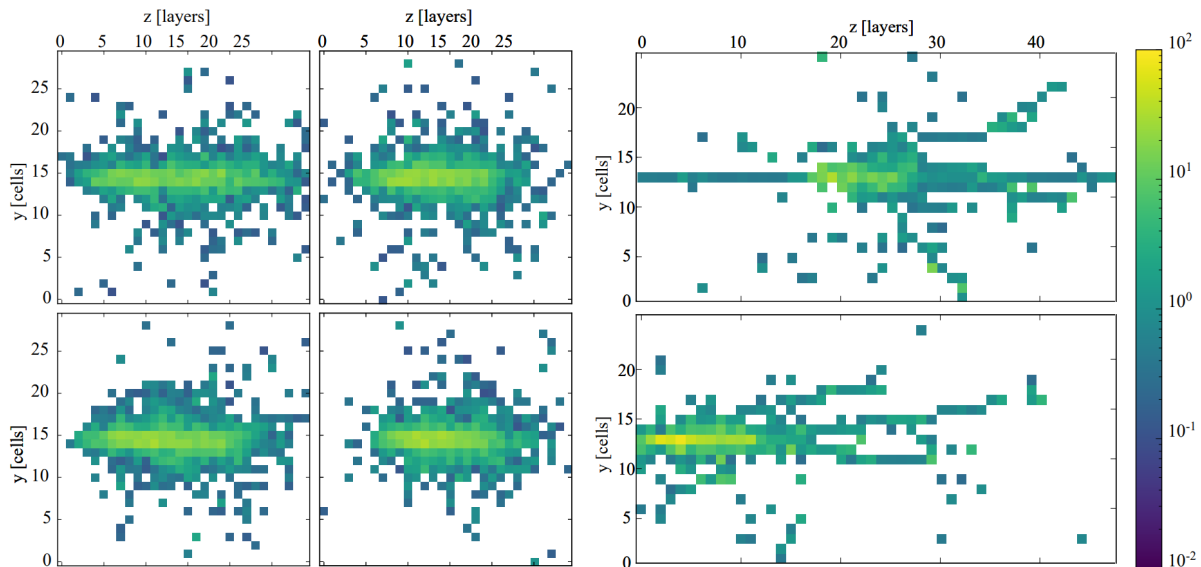


Figure 2.5: Comparison of four electromagnetic showers (left) and two hadronic showers (right). Note the uniform nature of the electromagnetic showers compared to the varied shape of the hadronic showers. Figure taken from [3].

## 2.3 Hadronic Showers

Hadron material interactions can be split into hadronic interactions and electromagnetic interactions. For charged hadrons, electromagnetic interactions behave similarly to the heavy charged particle case described previously. Neutral hadrons initially do not undergo electromagnetic processes, however, the hadronic interactions of neutral hadrons tend to produce charged hadrons, which can interact electromagnetically.

Hadronic interactions occur between the energetic hadrons and the nuclei of the material. This can take several forms, such as *spallation*, where the hadron is inelastically scattered on the protons and neutrons of the nucleus. This results in the production of several pions and depending on the energy transferred, even parts of the nucleus can be ejected. The nucleus itself is left in an excited state, leading it to either undergo *fission*, where the nucleus splits, releasing more energy in the process, or leading to nuclear *evaporation*, where the nucleus radiates off the excess energy as  $\alpha$ ,  $\beta$  or  $\gamma$  radiation. The produced pions and ejected nucleus fragments originating from the spallation can be energetic enough to cause further hadronic interactions, which again results in more particles being produced. This process continues until the energy of the resulting particles is insufficient to form new pions, resulting in what is called a hadronic shower. The remaining energy of particles that can no longer produce pions is lost either through radiation or by being captured into a nucleus.

Unlike in the rather deterministic outcomes of Bremsstrahlung or pair production, the exact number, and type of particles produced by nuclear interaction varies greatly. This leads to hadronic showers having significantly more diverse shapes than electromagnetic showers. Additionally, the number of interactions that result in secondary particles is significantly smaller in electromagnetic showers, making the statistical fluctuations in these few processes more im-

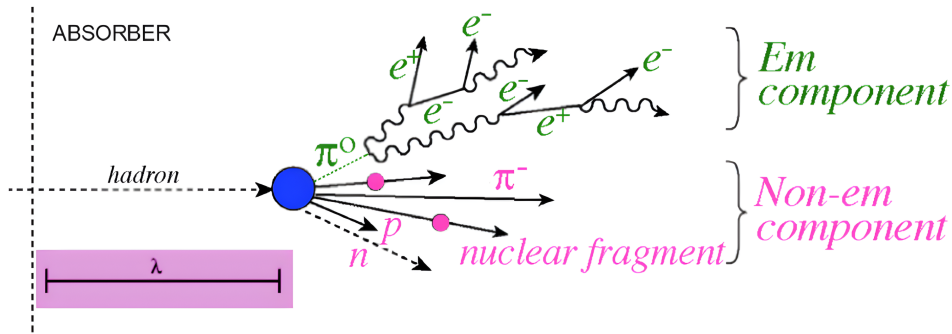


Figure 2.6: Schematic depiction of a hadronic shower. Note that the shower contains both hadronic and electromagnetic components. Figure taken from [90].

partful. Figure 2.5 compares four electromagnetic photon showers (left) with two hadronic pion showers (right). This both illustrates the uniformity of the photon showers and the variation of the pion showers.

Further, hadronic showers tend to also have an electromagnetic component. This is caused by neutral pions and  $\eta$ -mesons decaying into a pair of photons, each of which can initiate an electromagnetic sub-shower, as indicated in Figure 2.6, leading to further complexity in the shower profile.

The characteristic length scale of a hadron shower is the average distance traveled by a particle before undergoing a hadronic interaction. This is known as the *nuclear interaction length*  $\lambda_i$  and can be approximated as [88],

$$\lambda_i = (20A^{0.4} + 32) \frac{\text{g}}{\text{cm}^2} . \quad (2.3.1)$$

The nuclear interaction length tends to be significantly larger than the radiation length. This means that the volume to capture a hadronic shower needs to be significantly larger than what is required for an equally energetic electromagnetic shower.

## 2.4 Calorimeter Types

The task of a calorimeter is to capture and record particle showers in order to determine the energy of the particle that caused the shower. Performing this task for both hadronic and electromagnetic showers simultaneously is exceedingly difficult due to the vastly different sizes of the two shower types. Therefore, this task is divided up into two types of sub-detectors, the electromagnetic calorimeter (ECAL) and the hadronic calorimeter (HCAL). These two calorimeters are arranged so that a particle originating from the collision point first passes through the ECAL. The size of the ECAL is designed to be sufficient to capture EM showers, but small enough that hadrons will pass through largely without interacting. These hadrons then go on to deposit their energy in the significantly larger HCAL.

There exists a variety of methods for measuring the energy deposited by the showers, such as scintillators combined with photo multipliers, liquid argon, gas mixtures, or silicon diodes [88].

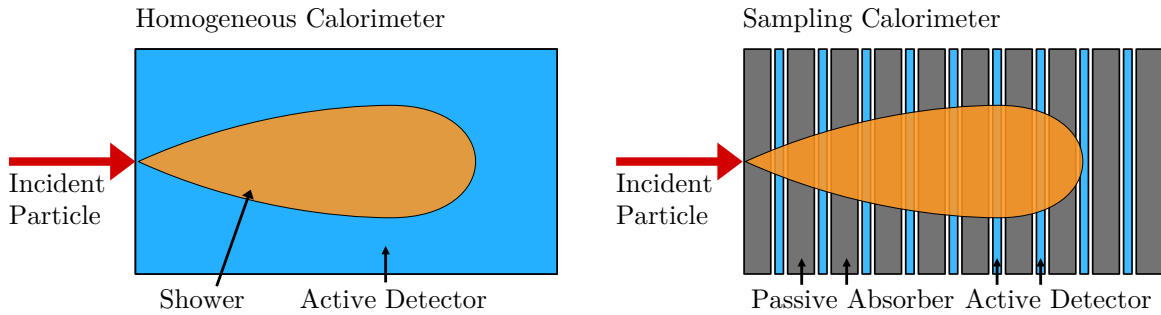


Figure 2.7: Comparison of homogeneous and sandwich sampling calorimeter approaches.

All of these approaches rely on recording the free charges caused by ionization processes. This means that only energy lost via ionization can be directly recorded. However, since the average ratio between ionization and other interactions is known, one can use the measured ionization energy to calculate the full shower energy.

Using these methods, there are two design approaches to building a calorimeter. In a *homogeneous calorimeter*, the entire volume is made up of active detector material. Commonly used are scintillating crystals made of dense materials such as the lead tungstate ( $PbWO_4$ ) used in the CMS ECAL [91]. A *sampling calorimeter* uses a combination of active detector material and dense passive absorber material. A common construction approach for sampling calorimeters is sandwich calorimeters, where the passive and active are arranged in alternating layers. Figure 2.7 illustrates the difference between a homogeneous calorimeter and a sandwich-type sampling calorimeter. Examples of the sampling design are the CMS High Granularity calorimeter (HGCal), a planned upgrade for the High Luminosity LHC, [92] and the ILD calorimeters that will be discussed in more detail in a separate section. The biggest difference between the sampling and homogeneous approach is that in a sampling calorimeter the majority of the energy is deposited within the absorber, and can therefore not be directly measured. Therefore, one has to use the fraction of energy that is recorded in the active material to infer the full shower energy. The fraction between the full energy  $E_{\text{full}}$  and the energy in the active sections  $E_{\text{active}}$  is known as the sampling fraction  $f_{\text{sampling}}$ , defined as

$$f_{\text{sampling}} = \frac{E_{\text{active}}}{E_{\text{full}}}. \quad (2.4.1)$$

A small sampling fraction can present a source of uncertainty in the measurement, which is a downside of a sampling calorimeter. However, sampling calorimeters have several advantages. The passive absorber can be significantly denser than an active material, such as tungsten ( $19.3 \frac{\text{g}}{\text{cm}^3}$ ) which has a significantly higher density than lead tungstate ( $8.2 \frac{\text{g}}{\text{cm}^3}$ ). This allows a sampling calorimeter to be more compact than a comparable homogeneous calorimeter. Further, the cost of absorber material tends to be lower than that of active material, making a sampling calorimeter more cost-effective. For this reason, nearly all HCALs are designed as sampling calorimeters, as their large volume requirement makes a homogeneous approach too costly, and the inherent fluctuations of hadronic showers make the fluctuations induced by the sampling

approach less impactful. Finally, the sampling setup allows the independent readout of the individual layers, thereby giving insight into the spatial structure of a shower in a way not possible with a homogeneous setup.

A further concept to mention is *high granularity* calorimeters. These calorimeters employ a large number of individual readout channels to produce a high-resolution recording of a calorimeter shower. The current approach to constructing high granularity calorimeters uses a sampling calorimeter setup, where each active layer is made up of a multitude of individual pixel detectors.

High granularity sampling calorimeters allow for an even better spatial resolution as the exact positions of energy depositions can be determined more precisely. A high spatial resolution is required for the particle flow approach [80]. Under this paradigm, one attempts to fully reconstruct every single particle in an event. To this end, one needs to be able to separate individual calorimeter showers in cases where multiple particles hit the detector at similar positions. This separation is only feasible for a sufficiently granular calorimeter. Additionally, high granularity calorimeters can be used to pile up rejection.

Since the main task of a calorimeter is to precisely measure the energy of a particle, one of the most important properties of a calorimeter is its energy resolution. The resolution is defined as the ratio

$$\text{resolution} = \frac{\sigma_E}{E}, \quad (2.4.2)$$

where  $E$  is the particle energy and  $\sigma_E$  is the uncertainty on the measured energy. In practice,  $\sigma_E$  can be understood as the standard deviation one expects when repeatedly measuring particles with the same energy.

Calorimeter resolution can be decomposed into several contributions [93]:

$$\left(\frac{\sigma_E}{E}\right)^2 = \left(\frac{a}{\sqrt{E}}\right)^2 + \left(\frac{b}{E}\right)^2 + c^2. \quad (2.4.3)$$

The first term  $\frac{a}{\sqrt{E}}$  corresponds to stochastic fluctuations intrinsic to particle showers. Since the development of a shower is not a deterministic process, the exact processes by which the shower particles lose energy will vary from shower to shower. In a sampling calorimeter, there are additional fluctuations, as the energy depositions can either take place in the active or passive parts of the calorimeter. These fluctuations scale approximately with  $\sqrt{N}$ , where  $N$  is the number of particles in the shower. Since  $N$  is proportional to the total energy of the particle, this results in an overall scaling of  $\frac{\sqrt{E}}{E} = \frac{1}{\sqrt{E}}$ .

The second contribution,  $\frac{b}{E}$  is the so-called noise term. It originates from the intrinsic noise of the sensors and readout electronics of the calorimeter. This term is constant, regardless of the energy of the particle that is measured, meaning its relative contribution drops for more energetic particles.

The final term,  $c$  presents a constant contribution to the resolution that does not become less relevant for higher particle energies. This term stems from imperfections in the detector, such as inhomogeneities or imperfect calibrations. These effects result in a relative smearing that is more impactful at higher particle energies, leading to the constant contribution to the ratio  $\frac{\sigma_E}{E}$ .

## 2.5 Calorimeter and Shower Simulation

### GEANT4 Full Simulation

The state-of-the-art software for simulating calorimeter showers is the GEometry ANd Tracking toolkit (GEANT4) [29]. GEANT4 is a first principle, full simulation setup, meaning it aims to model the paths and interactions of every particle in a shower using a wide range of interaction models.

A detector in GEANT4 is implemented as a combination of volumes, with each volume consisting of a specific material. Individual particles within the detector are simulated using a step-based model. At each step, the possible interactions for a particle are considered. These interactions are characterized using the mean free path length  $\lambda$ . The probability  $p$  of interacting after a traversed distance  $l$  is given by the exponential relation

$$p(l) = e^{\int_0^l \frac{1}{\lambda(l)} dl} . \quad (2.5.1)$$

The integral in the exponent accounts for varying mean free path lengths at different points along the trajectory of the particle, for example, caused by the particle losing energy along the way or traversing different materials.

For most interactions, the mean free path length in a section of material can be expressed as

$$\frac{1}{\lambda} = \rho \sum_i \frac{x_i \sigma_i}{m_i} , \quad (2.5.2)$$

where the index  $i$  corresponds to various isotopes contained within the material with overall density  $\rho$ .  $\sigma_i$  is the cross section of the interaction with a specific isotope,  $m_i$  is the mass of the isotope, and  $x_i$  describes the fraction of the material made up of the isotope.

In order to randomly sample path lengths of processes, Equation (2.5.1) can be rewritten as

$$p(l) = e^{-n_\lambda} , \quad \text{with} \quad (2.5.3)$$

$$n_\lambda = \int_0^l \frac{1}{\lambda(l)} dl . \quad (2.5.4)$$

This allows for the sampling of  $n_\lambda$  from a simple exponential distribution, which is achieved by drawing a sample  $\eta$  from a uniform distribution between 0 and 1 and setting

$$n_\lambda = -\ln \eta . \quad (2.5.5)$$

The sampled values for  $n_\lambda$  can then be used to calculate the corresponding distance  $l$  before the interaction occurs. This distance is separately determined and sampled for every possible process. The shortest distance among all possible interactions determines the current step size and what process the particle undergoes at the end of this step. An additional constraint to the step size is that a single step is not allowed to cross a boundary between detector volumes. In detectors that comprise a large number of individual materials and components, such as highly

granular sampling calorimeters, this constraint can significantly reduce the average step size, meaning the overall simulation requires more steps to be taken.

At the next step, the  $n_\lambda$  values are modified based on the previous step size, and the algorithm is applied again to select the next process. This repeats until the energy of the particle drops below a particle type specific threshold, after which the remaining energy is deposited in the current volume and the simulation of this particle is terminated.

The exact processes that are considered for each particle are defined by an extensive and well-refined list of possible interactions implemented in GEANT4. Various settings are available that make use of different underlying theoretical models to simulate the interactions. The exact choice of settings is highly dependent on the specific application and energy range that one intends to run simulations.

The development of electromagnetic showers and other electromagnetic interactions are well described by a standard package of EM processes. Hadronic interactions are more difficult to cover with a unified list, due to their more complex and varied interactions. Therefore, several options are available to simulate hadronic showers. The two *physics lists* used for hadron shower simulations in this thesis are the *QGSP\_BERT\_HP* and *FTFP\_BERT\_HP* lists. Both lists make use of the Bertini cascade model [94] (BERT) for low energies and employ the High Precision neutron setting (*HP*). This setting lowers the energy threshold at which neutron simulations are terminated, resulting in more accurately described interactions of low-energy neutrons.

The two lists differ in how they model the high energy regions, where *QGSP\_BERT\_HP* makes use of the quark-gluon string precompound modeling [95] and *FTFP\_BERT\_HP* uses the Fritiof precompound model [96].

## Fast Simulation Approaches

The complete simulation of every particle in a calorimeter shower is both time-intensive and computationally expensive. Further, the interactions of a given particle are dependent on previous interactions the particle underwent. Therefore, the simulation has to be done sequentially and cannot be easily parallelized.

To alleviate the strain shower simulation puts on computation resources in particle physics, several approaches for less accurate, but significantly faster simulation, known as FastSim, have been proposed and developed.

Methods based on generative ML models are promising candidates and will be discussed in more detail in subsequent chapters. However, there also exists a series of FastSim not based on ML.

DELPHES [97] is a software framework capable of quickly simulating entire detectors. Rather than modeling individual showers, DELPHES assigns the energy of particles that arrive in the calorimeter to a single ECAL and a single HCAL cell at the impact position of the particle. The choice of the relative fraction between ECAL and HCAL is based on the particle type. For example, electrons and photons deposit their entire energy in the ECAL, while stable hadrons deposit their energy completely in the HCAL. To model the imperfect resolution of a real calorimeter, the actually deposited energy is smeared by a factor defined using the calorimeter resolution. This results in an exceedingly fast simulation; however, DELPHES requires the energy resolution of a calorimeter to be known in advance and therefore cannot be used for first-principle simula-



tions. Additionally, DELPHES only models the total energy, but not the shower shapes, making it difficult to use in conjunction with particle flow.

GFLASH [98,99] is a parameterized approach to shower simulation based on GEANT4. It uses longitudinal and transversal energy profiles derived from showers fully simulated in GEANT4 and uses MC methods to place energy depositions in the simulated calorimeter such that these profiles, as well as additional internal correlations, are accurately reproduced. This approach is significantly faster than a full simulation of every particle. This parameterized approach can be used in conjunction with any previously listed GEANT4 setting, so long as data simulated using this setting is used as the basis of the parametrization.

*Frozen Showers* [100] describes an approach where a library of showers caused by low energetic particles is simulated beforehand using a full simulation. During the main simulation, the Frozen Showers approach uses either a FullSim or a parameterized approach to deal with highly energetic particles. However, once a sub-particle falls below a certain threshold, the simulation of this particle is terminated and one of the pre-generated showers with appropriate energy, angle, and position is placed at the position of the particle. Thereby, the Frozen Showers method skips having to simulate the low energetic particles that make up a majority of shower particles.

A similar approach had already been pioneered by the H1 collaboration [101], which used a library of pre-simulated showers for fast simulation of complete showers, rather than sub-showers.

## 2.6 ILD Calorimeters

The ILD [59] is one of the proposed detectors for the ILC. The general features of both the ILD and ILC have been introduced in Section 1.4. This section specifically focuses on the calorimeter designs proposed for the ILD. As the ILD is specifically designed and optimized for the particle flow approach, both its calorimeters require sufficient resolution to allow for the separation of overlapping showers.

The CALorimeter for LInear Collider Experiment (CALICE) [62] group works to design, build and test particle flow optimized calorimeters. Work done by CALICE has also found application in the CMS high granularity calorimeter upgrade [102].

### ILD ECAL

There are two proposed designs for the electromagnetic calorimeter of the ILD. The SiW-ECAL [103] is a sampling calorimeter with tungsten layers as absorbers and active silicon detectors. The ScW-ECAL [104] design similarly uses tungsten absorber layers but uses scintillators instead of silicon detectors.

The design used in this thesis is the SiW-ECAL. It consists of a total of 30 active and 30 passive layers, arranged alternately. The active silicon layers have a thickness of 0.525 mm and are made up of individual cells with a size of 5 mm  $\times$  5 mm, where each cell is individually read out. The first 20 passive absorber layers have a thickness of 2.1 mm, while the remaining 10 have a thickness of 4.2 mm. This allows for a high longitudinal resolution in the first parts of the calorimeter, while still providing enough total material to contain the majority of an electromagnetic shower. The borders between two sensor cells present possible blind spots. To alleviate this problem, the individual cells are staggered between layers, thereby preventing particles from repeatedly passing through the gaps between cells.

## ILD HCAL

The two design proposals for the hadronic calorimeter design are the Analogue Hadron Calorimeter (AHCAL) [63] and the Semi Digital Hadron Calorimeter (SDHCAL). For this work, we focus on the AHCAL design. Similar to the ECAL, the AHCAL is a highly-granular sampling calorimeter. It uses stainless steel layers with a thickness of 17.2 mm as the absorber material and 3 mm thick scintillators in combination with silicon photomultipliers as the active detector elements. Each scintillator tile has a size  $3\text{cm} \times 3\text{cm}$  and the design features 48 active layers.

A prototype version of this calorimeter with 38 layers was tested at the Super Proton Synchrotron (SPS) at CERN [105].

## ILD Calorimeter Simulation

The construction and planned use case of the ILD calorimeter system put it at an intersection of several factors that make MC simulations for the calorimeters challenging. On the one hand, the segmented nature of the highly granular sampling calorimeters forces GEANT4 simulations to operate with reduced step sizes, resulting in extensive per-event simulation times. On the other hand, the high precision measurements planned at the ILC require a vast amount of measured events, the analysis of which will, in turn, require a vast amount of MC data.

This combination of lengthy simulation times and large MC data requirements make the ILD calorimeter system a prime beneficiary of the generative FastSim approaches explored in this work.

## Chapter 3

# Machine Learning

Machine learning (ML) is a general description of algorithms designed to automatically improve and adapt through the usage of data. The field covers a wide range, from simple boosted decision trees to highly complex deep Neural Networks.

The general principles of ML have been under investigation for quite some time, however, their need for significant computational resources presented a significant hurdle. Recent advances in computational hardware, specifically the arrival of highly parallel graphical processing units (GPUs) have enabled the widespread use of ML in both science and industry.

ML can broadly be separated into two subcategories, supervised learning, unsupervised learning, and Reinforcement learning. Supervised methods work on labeled data, meaning each point in the data set has a corresponding label that describes some property of the data point. A set of images with a label designating the image content is an example of such a labeled data set. The most common supervised learning tasks are classification tasks, where a model aims to assign an appropriate label to a given data point, and regression tasks where the model tries to predict a specific value associated with that data point. Unsupervised methods use unlabeled data. Such methods are largely designed to learn underlying structures within the data set. Common applications are anomaly detection and generative models, which will be the main focus of this chapter. Reinforcement learning features an agent that performs decisions based on a set of circumstances in order to achieve a specific goal. The agent is assigned a score based on how well it performs at reaching this goal and is trained to maximize the score it receives. Reinforcement learning is most commonly used in applications with abstract goals that cannot easily be expressed mathematically. Deep learning is a sub-field of ML concerned with deep neural networks.

This Chapter covers the underlying methodology of ML, starting with an introduction to the principle of gradient descent in Section 3.1. Sections 3.2 and 3.2 cover commonly used ML optimization algorithms and loss functions, respectively. General challenges and potential problems encountered in ML are discussed in Section 3.4. In Section 3.5 neural networks are introduced, and several layer types used to construct neural networks are presented. Finally, Section 3.6 specifically covers activation functions used in neural networks.

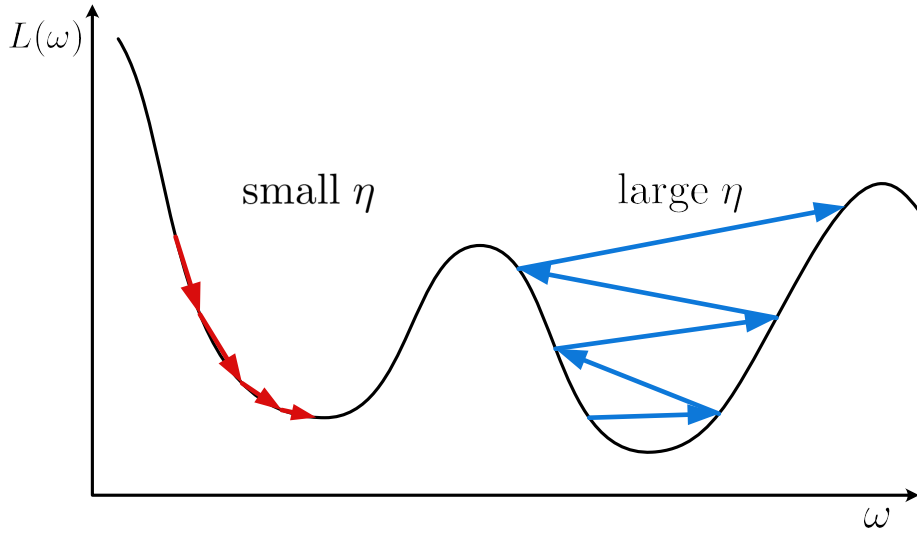


Figure 3.1: Potential risk of too small and too large learning rates. Left: optimizer with small learning rate converging to a local minimum, right: optimizer with large learning rate leaving the global minimum.

### 3.1 Gradient Descent

The underlying concept of both machine and deep learning is to define a model  $f(x)$ , with  $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$ , and to iteratively modify this model until it describes a given data-set  $X = \{\mathbf{x}_k\}_{k=0}^K$ , where  $K$  is the total number of data points, and  $\mathbf{x}_k \in \mathbb{R}^N$ . Such a model can be understood as a nonlinear function mapping from an  $N$ -dimensional input space to an  $M$ -dimensional output space. This function is defined by an arbitrary number of parameters  $\omega$ , which are optimized to facilitate the iterative improvement of the model [106]. This optimization takes the shape of minimizing a so-called *objective* or *loss* function  $L$ . Some specific loss functions will be explained in more detail in the following section, for now, a loss function can be understood as a function that measures the performance of the model, such that a small loss corresponds to better performance. Therefore, the exact value of the loss will be dependent on the parameters of the model and the training data.

The most common approach used to minimize a loss function of a model in ML is gradient descent. To this end, one calculates the gradient of the loss function with respect to the model parameters,

$$\nabla_{\omega_i} L(\omega, X) = \frac{\partial L(\omega, X)}{\partial \omega_i}, \quad (3.1.1)$$

where  $\omega_i$  is the  $i$ -th parameter in  $\omega$ . Note that gradient descent requires a differentiable loss function. This gradient can be seen as the direction in the parameter space, in which the loss function increases, therefore by modifying each parameter in the opposite direction, the loss will be reduced. One such gradient descent update step can be written as

$$\omega_{t=j+1} = \omega_{t=j} - \eta \nabla_{\omega_{t=j}} L(\omega, X). \quad (3.1.2)$$

Here  $\omega_{t=j}$  are the model parameters after  $j$  gradient descent steps, and  $\eta$  is a parameter that determines the step size of each update step.  $\eta$  is commonly referred to as the learning rate. The learning rate has significant effects on how well a gradient descent process is able to find a global loss minimum. A too large learning rate runs the risk of overshooting and missing the global minimum, as illustrated by the blue line in Figure 3.1. However, a too small learning rate can end up stuck in a local minimum, also causing the global minimum to be missed, as illustrated by the red line in Figure 3.1. Furthermore, a small learning rate means the optimization makes less progress per step, thereby increasing the time the model takes to converge.

## 3.2 Optimizers

The potential problems of gradient descent have led to the development of several, more sophisticated optimizing methods derived from gradient descent to help alleviate these problems.

### Momentum

One step of standard gradient descent is calculated using the entire data set  $X = \{x_k\}_{k=0}^K$ . This can be problematic for complex models and large data sets, as calculating and storing the gradients for each model parameter may require more computational resources than are available. Stochastic gradient descent (SGD) [107] modifies the approach by splitting the data into a set of disjoint subsets  $X_j = \{x_k\}_{k=b(j-1)}^{bj}$  called minibatches, with  $j \in \{1, \lceil \frac{K}{j} \rceil\}$ . The SGD steps are then performed independently and consecutively on all batches.

This allows for more complex models, as only a fraction of the full data set needs to be evaluated per step. Additionally, the exact composition of the batches is random, which introduces some statistical fluctuations to the gradients of the different batches. These fluctuations reduce the risk of the optimizer converging to a local minimum, as a parameter set that corresponds to a local minimum for one batch will not be a minimum for a different batch, thereby allowing the SGD algorithm to escape local minima.

### Momentum

Both gradient descent and SGD will encounter problems in so-called ravines [107], regions in gradient space where there is a steep gradient valley in one direction, but only a gentle slope in another. In such as case the SGD will follow the largest gradients, causing it to bounce back and forth between the steep valley walls while making little progress along the slope. The left-hand side of Figure 3.2 illustrates this effect.

One approach to solve this is the addition of a momentum contribution, that adjusts the current trajectory based on the previous step. If a standard SGD update step is defined as

$$\omega_{t=j+1} = \omega_{t=j} + \Delta\omega_{t=j} \text{ with} \quad (3.2.1)$$

$$\Delta\omega_{t=j} = -\eta \nabla_{\omega_{t=j}} L(\omega, X), \quad (3.2.2)$$

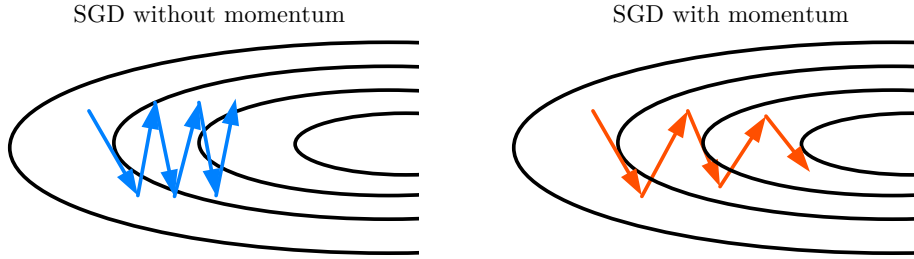


Figure 3.2: Behavior of SGD with and without momentum in a *ravine*. Black lines indicate contours in the loss function. Without momentum, the SGD will bounce back and forth, while the momentum term guides the optimizer down the slope.

then we can expand this to include a momentum term by adding a scaled version of the previous update vector  $\Delta\omega_{t=j}$  to the current one, resulting in

$$\omega_{t=j+1} = \omega_{t=j} + \Delta\omega_{t=j} \text{ with} \quad (3.2.3)$$

$$\Delta\omega_{t=j} = \gamma\Delta\omega_{t=j-1} - \eta\nabla_{\omega_{t=j}}L(\omega, X), \quad (3.2.4)$$

where  $\gamma$  is the scaling factor dictating the strength of the momentum.

This helps both to dampen the oscillations and to enhance the optimizer movement along the gentle but consistent downward slope. Overall, this leads to a fast convergence, as indicated in the right illustration in Figure 3.2.

## ADAM

Adaptive Moment Estimation (ADAM) [108] is the current state-of-the-art optimization algorithm. ADAM is based on SGD but expands it by two running averages over the gradients and the squared gradients denoted as  $m_{t=j}$  and  $v_{t=j}$  respectively [107]. These terms are defined by

$$m_{t=j} = \beta_1 m_{t=j-1} + (1 - \beta_1)\nabla_{\omega_{t=j}}L(\omega, X) \quad (3.2.5)$$

$$v_{t=j} = \beta_2 v_{t=j-1} + (1 - \beta_2)[\nabla_{\omega_{t=j}}L(\omega, X)]^2. \quad (3.2.6)$$

Here  $\beta_1$  and  $\beta_2$  are hyperparameters that define how fast the running average decays. The suggested default values are  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . The initial values of the two terms are set to be  $m_{t=0} = 0$  and  $v_{t=0} = 0$ . Depending on the decay rates both  $m_{t=j}$  and  $v_{t=j}$  can be biased toward zero. To account for this one introduces the corrections

$$\hat{m}_{t=j} = \frac{m_{t=j}}{1 - \beta_1} \quad (3.2.7)$$

$$\hat{v}_{t=j} = \frac{v_{t=j}}{1 - \beta_2}. \quad (3.2.8)$$

The gradient update step is then performed using these two parameters,

$$\omega_{t=j+1} = \omega_{t=j} - \frac{\eta}{\sqrt{\hat{v}_{t=j} + \epsilon}} m_{t=j} . \quad (3.2.9)$$

This means the actual update direction is given by the running average over the past updates. Similar to a momentum term this allows ADAM to quickly traverse steady slopes in gradient space. Additionally, the learning rate is scaled by the squared mean over the gradient updates, which can be understood as their variance. This allows the optimizer to effectively have a small learning rate in gradient regions with significant fluctuations, while at the same time giving the optimizer a large learning rate in regions without much variance. The combination of these two factors allows ADAM to reliably and quickly converge. Additionally, the adaptive scaling of the learning rate reduces the impact of  $\eta$  as a hyperparameter, thereby enabling stable training without having to fine-tune the learning rate.

### 3.3 Loss Functions

A loss function also called cost or objective function is the metric by which a network is optimized. In principle, any function can be a loss function, so long as a smaller loss value corresponds to improved network performance. Furthermore, a loss function needs to be differentiable, as the gradient descent approach and its derivatives require the calculation of the gradient  $\nabla_{\omega_{t=j}} L(\omega, X)$ .

There exists a vast amount of loss functions, tailored to specific problems and applications, such that an exhaustive list would be beyond the scope of this work, however the most commonly used loss functions in classification and regression tasks will briefly be outlined. Loss functions for generative models will be discussed in more detail in Chapter 4.

#### Mean Squared Error

In a supervised classification or regression task, one has a set of data  $X = \{x_k\}_{k=1}^K$  and a set of so-called labels  $Y = \{y_k\}_{k=1}^K$ . Each data point  $x_k$  has a specific label  $y_k$  associated with it. These labels can describe any sort of property of the data. A common example for a classification task is to have a data set containing images of either dogs or cats and a set of labels that indicates which animal is depicted in the images. Meanwhile, a more physics-specific example of a regression task could be a data set of calorimeter showers, with labels describing the energy of the incident particle. The labels are usually encoded into numbers, so an ML model can easily interact with them. For the cat and dog images, this would for example mean assigning the value 0 to the “cat” label and the value of 1 to the “dog” label, while in the shower example the labels could be the particle energy in GeV.

The actual classification or regression task now consists of creating a model  $f(x)$ , such that  $f$  predicts the correct label for a given data point. In order to optimize the model, one needs a loss function that measures the accuracy of the model predictions.

A straightforward way of achieving this is a mean squared error loss (MSE) function. This loss calculates the average squared difference between the network predictions and the true labels values, given by the expression

$$L_{\text{MSE}} = \frac{1}{K} \sum_{k=1}^K [f(x_k) - y_k]^2. \quad (3.3.1)$$

The output of the MSE becomes smaller, the closer the model predictions are to the true values, and can easily be differentiated.

### Cross Entropy

Cross entropy (CE) is a loss function specific to classification tasks with exactly two classes, such as the animal images example. CE is defined as

$$L_{\text{CE}} = \frac{1}{K} \sum_{k=1}^K \left( -y_k \log f(x_k) - (1 - y_k) \log(1 - f(x_k)) \right). \quad (3.3.2)$$

It is important to note that CE is only defined for networks with outputs constrained to  $[0, 1]$ .

To understand Equation (3.3.2) one can break it down into two cases. The first is  $y_k = 0$ , in this case the first term becomes 0 and only  $-\log(1 - f(x_k))$  remains. This expression is maximized as  $f(x_k)$  approaches 1 and minimized as it approaches 0. Thereby the loss pushes  $f(x_k)$  toward the correct label. In the second case where  $y_k = 1$  only the first expression  $-\log f(x_k)$  remains relevant. Similar to the previous case, this pushes  $f(x_k)$  toward the, in this case correct, label of 1.

## 3.4 Challenges of Machine Learning

There are several difficulties one may encounter when training or otherwise optimizing an ML model. Some of the more common problems will now briefly be covered.

### Overfitting

Overfitting [109] describes the case where a model attempts to perfectly fit a given data set, to the point of modeling statistical fluctuations in the data set. One example would be the model learning every training point by heart, rather than describing the underlying distribution from which the data points originate. This case is illustrated in Figure 3.3 where the overfitting model exactly describes every data point but shows a large deviation from the true distribution.

The biggest problem of an overfitting model is that a model trained on a specific data set will only accurately describe the points of that training data set. Applying the model to any other data points from the same distribution will result in significantly worse performance. Overfitting is often linked to having an overparameterized model, where the model has more available parameters than are required to perform the task at hand. However, recent research [110] suggests that this is not universally true, and significantly overparameterized models can be less prone to overfitting.

There exist several ways to prevent or reduce overfitting. The most common approach is to split the data into a training and a validation subset. While the model is trained on the training set, its performance is regularly evaluated in the validation set. If the training loss and validation loss start to diverge, it is an indication of overfitting and the training can be stopped early.



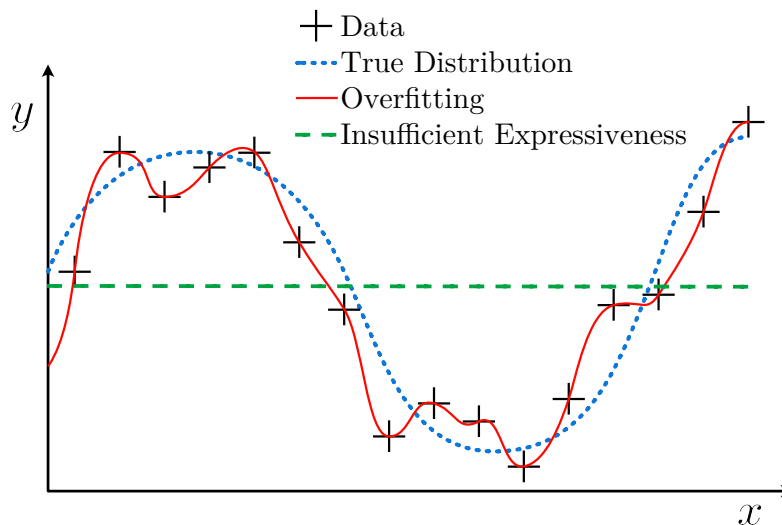


Figure 3.3: Illustration of the effects of overfitting (red, solid) and insufficient model expressiveness (green, dashed). In both cases, the resulting description differs significantly from the true distribution (blue, dotted).

### Insufficient Model Capability

Having a model with insufficient descriptive capacity, also known as insufficient expressiveness, can be seen as the opposite of overfitting. While overfitting describes a case where the model can learn every data point by heart, insufficient expressiveness leads to the model being unable to fully capture the underlying distribution in its entirety. Figure 3.3 again illustrates this case.

Insufficient model expressiveness often results in unsatisfactory model performance. This makes it a hard-to-spot problem, as it can be difficult to determine whether the model is not capable enough or the training procedure is simply sub-optimal.

### Vanishing Gradient

Training an ML model through gradient descent requires the model to be differentiable. However, even differentiable models can still have an effective gradient of 0. In such cases, the optimizer no longer modifies the model parameters and the training becomes stuck in its current state.

There exist a variety of causes that result in a gradient of 0. Vanishing gradients caused by local extrema in the loss function are generally easy to resolve through the use of a momentum term in the optimizer or SGD. However, vanishing gradients caused by the model construction itself can be more difficult to resolve.

## 3.5 Neural Networks

Neural networks form the basis of deep learning. In essence, a neural network is a nonlinear function mapping from an  $N$ -dimensional input space to an  $M$ -dimensional output space. The precise nature of this mapping is dependent on the structure of the network, also known as its architecture. For practical reasons, neural networks are commonly constructed as the combination

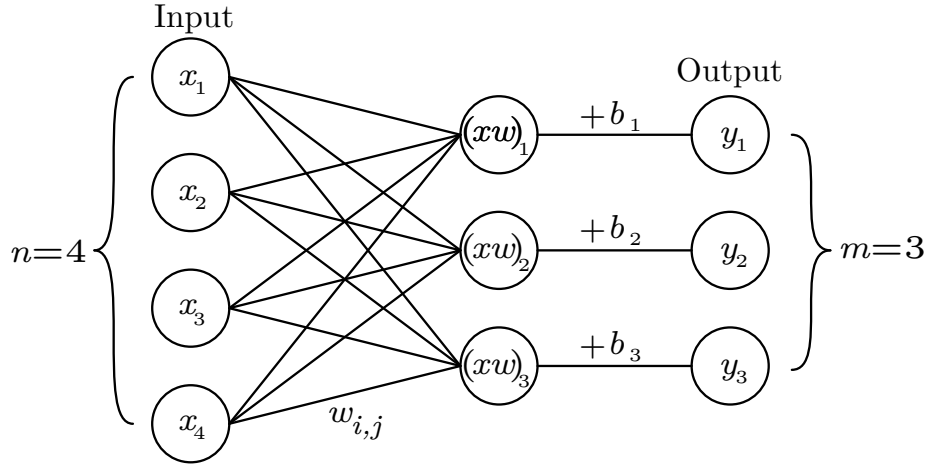


Figure 3.4: Dense neural network layer with a bias term  $b$ , 4 input nodes, and 3 output nodes.

of several so-called layers. A network is referred to as *deep* if it contains multiple layers. These layers can themselves be understood as functions with variable input and output dimensionality that, when applied in succession, form the complete network. The first layer in a network is called the input layer, the last layer is the output layer and any intermediary layers are known as hidden layers. Several such layers that will be used throughout this thesis will be discussed in the following.

## Dense Layers

The basic building block of a neural network layer is the so-called neuron. Broadly modeled after its biological counterpart, such a neuron takes a series of inputs and transforms them into a scalar output. While the specifics of this transformation are arbitrary, dense layer neurons make use of a specific weighted sum transformation. The output  $y$  of this is the dot product of the input vector  $\mathbf{x} = \{x_1, \dots, x_n\}$  and a weight vector  $\boldsymbol{\omega} = \{\omega_1, \dots, \omega_n\}$

$$y = \sum_i^n x_i \omega_i + b. \quad (3.5.1)$$

The additional  $b$  is a biased term that offers greater flexibility to the transformation. Both  $\boldsymbol{\omega}$  and  $b$  are variable parameters of the transformation that are adjusted during the training procedure, also called trainable parameters.

A single neuron only provides a 1-dimensional output. However, several neurons with independent weight vectors can be applied to the same input vector, resulting in one output dimension per neuron. This can be expressed by replacing the output scalar with a vector  $\mathbf{y} = \{y_1, \dots, y_m\}$  and expanding the weight vector and bias term into a matrix  $\boldsymbol{\omega}_{n \times m}$  and vector  $\mathbf{b} = \{b_1, \dots, b_m\}$ . This modifies Equation (3.5.1) into

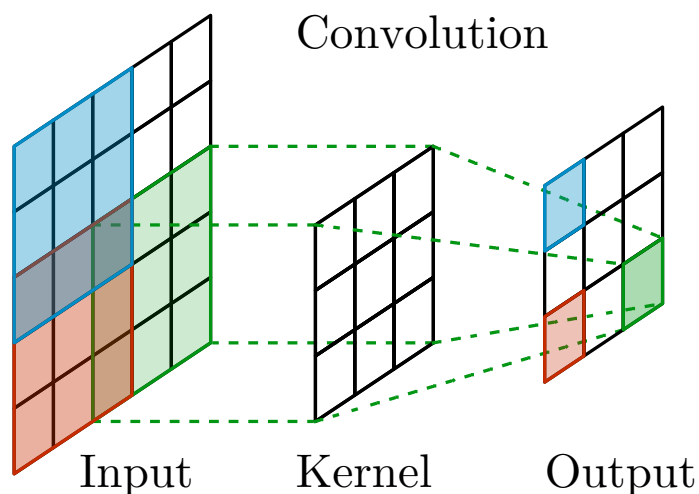


Figure 3.5: 2-dimensional convolutional operation with a  $3 \times 3$  kernel.  $3 \times 3$  regions of the input are multiplied element-wise with the kernel and summed over to obtain the output value.

$$y_j = \sum_i^n x_i \omega_{i,j} + b_j. \quad (3.5.2)$$

Figure 3.4 visualizes such a combination of three neurons with four inputs to each. Within this configuration, every input is connected to every output through a weight. For this reason, such a layer is referred to as a fully-connected layer or dense layer. The total number of trainable parameters of a dense layer is the sum of the number of elements in the weight matrix and of the elements in the bias vector, resulting in

$$N_{\text{params}} = n \times m + m = (n + 1) \times m. \quad (3.5.3)$$

### Convolutional Layers

Convolutional layers are based on convolutional filters used in image processing. The underlying principle is illustrated in Figure 3.5. The convolution filter is overlaid onto an input region of the same size as the filter. Now every entry in that region is multiplied with its respective counterpart in the convolutional filter and the results are summed up, resulting in a scalar output for this specific region. The filter is then moved along the first dimension of the image and the multiplication is performed once again at this new position. The distance by which the filter is moved is known as its stride. This scanning process continues until the whole image has been covered by the convolutional filter, resulting in a 2-dimensional map of scalar outputs called a feature map. A stride larger than one results in fewer overall steps in the convolution and therefore a smaller output size. This type of downsampling behavior can be utilized to reduce the too-large input sizes to more manageable ranges. Classical convolutional operations

in image processing make use of specialized kernels to extract specific features, such as horizontal or vertical edges, circular structures, or gradients. Convolutional neural network layers, on the other hand, do not have specific filters, but instead, use filters formed from trainable parameters. This allows the filter to be adjusted during training to extract desired features.

In general, it is not sufficient to extract one single feature from a given input. Therefore convolutional neural network layers have the capacity to handle multiple convolutional filters in parallel. Each of these filters is applied to the same input and each results in an independent output feature map. Further, the inputs to a convolutional layer can themselves have multiple feature maps. In the case of  $n$  input feature maps and  $m$  output feature maps, one first applies a total of  $m$  convolutional filters to each of the  $n$  input maps, resulting in  $n \times m$  total filters. The outputs of these filters are then summed in a way that ensures every sum has exactly one contribution from each of the  $n$  input maps, leading to the desired total of  $m$  outputs.

This enables the stacking of multiple convolutional layers. In such a setup the earlier convolutional layers can extract basic features from the input, which is then combined into more complex features by later layers.

When one applies a convolutional operation with a kernel size larger than one to an image, the resulting output will have a lower number of pixels compared to the input. This behavior can be undesirable depending on the application. For such cases, one can use so-called padding. This involves placing sufficient additional pixels around the input so that the output has the same number of pixels as the un-padded input. The precise content of the added pixels depends on the exact type of padding. The most common one, so-called zero-padding, uses zeros for this purpose.

As convolutions originated from image processing, their native input structure is matrices of pixels. However, the concept can also be adapted to vector- or higher-order tensor-like structures. Such convolutional layers are known in ML as 1D- or 3D-convolutions, respectively.

## Transpose Convolutional Layers

Transpose convolutions present an inversion of standard convolutional operations. Like convolutions, their standard applications are 2-dimensional images. However, while standard convolutions multiply individual entries of the input with individual kernel parameters and sum over the result, transpose convolutions multiply every individual input entry with the entire transpose convolutional kernel. Each of these multiplications results in an intermediate output with a size equal to that of the kernel. These intermediate matrices are then overlaid with a distance between each matrix equal to the stride of the transpose convolution. Figure 3.6 shows the principle of a transpose convolution. Similar to how a large stride can be used for downsampling in classical convolutions, a large stride in transpose convolutions results in an upsampled output. Compared to other upsampling approaches, transpose convolutions are not static, as the kernel is composed of trainable weights, allowing for wider flexibility in the upsampling.

Many other features of classical convolutions, such as multiple feature maps, padding, and extension to 1- and 3-dimensional data are also present for transpose convolutions.

## Dropout

Dropout [111] is an approach used to reduce overfitting and introduce redundancy into a neural network. This is achieved by giving each input to a layer a chance  $p$  to be turned off in any

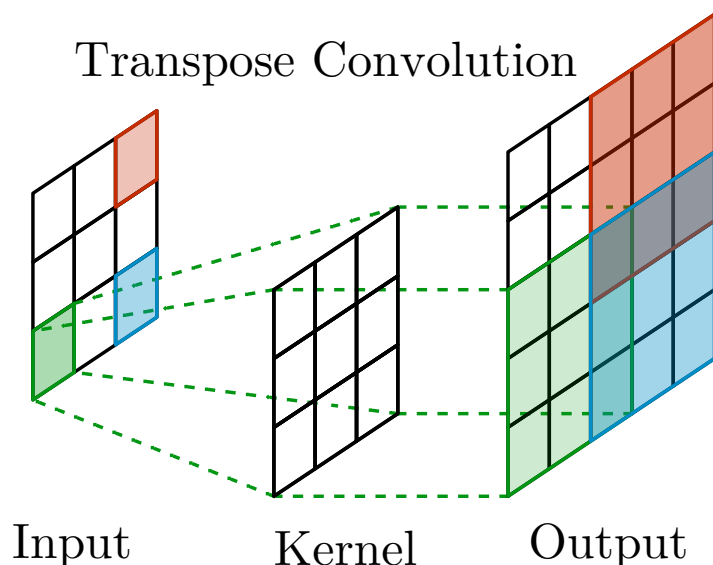


Figure 3.6: 2-dimensional transpose convolutional operation with a  $3 \times 3$  kernel. Individual inputs are multiplied with the kernel, resulting in a  $3 \times 3$  region in the output.

given training step. Therefore the network only has access to a randomly chosen subset of its intermediate values each time it is evaluated during training. This discourages the network from placing a too strong emphasis on specific intermediate or input features, as they will not always be available. Figure 3.7 shows the effect of dropout with  $p = 0.5$  on a fully connected network. In the left network, approximately half of the nodes are turned off, reducing the number of overall connections to one quarter.

Dropout is not a full layer, but rather a modification that is applied to other layers. In practice, the deactivation of nodes is done by masking out certain inputs to a layer and the outputs of the remaining layers are scaled by a factor of  $\frac{1}{1-p}$ . Generally, it is turned off when the network is evaluated after training, to ensure deterministic network outputs. The scaling factor applied during training ensures dropout can be switched off while keeping the intermediate network inputs consistent.

### Batch Normalization Layers

Many ML applications deal with a large range of possible values in their various input features. Depending on the specific data set certain input features may differ from other features by several orders of magnitude. Such discrepancies between input features are problematic. Therefore it is common practice to normalize input features to all lie within the same range, usually by normalizing their mean to 0 and their standard deviation to 1.

However, this problem is not limited to the network inputs. The inputs to intermediate layers can also have vastly different orders of magnitude. Batch normalization (or batchnorm) layers [112] are designed to address this. The input batch  $x$  to a batchnorm layer gets transformed according to the following rule

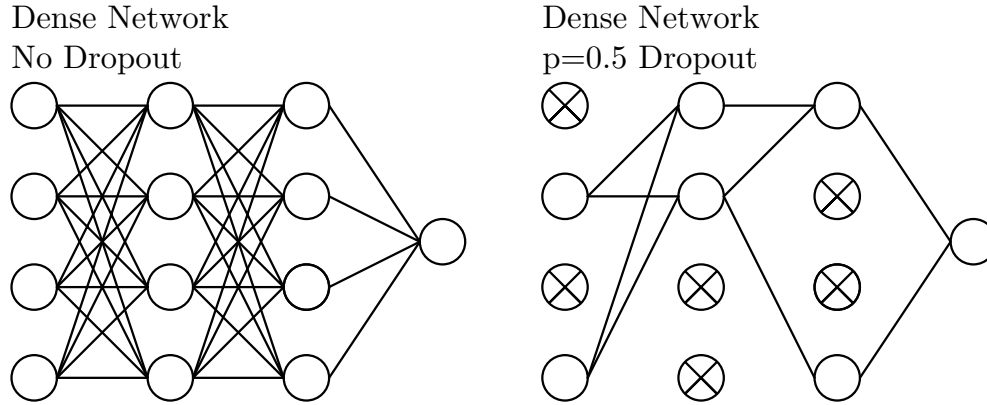


Figure 3.7: Demonstration of the effects of dropout using a 3 layer dense network. The nodes that are deactivated are chosen at random with a probability of  $p = 0.5$  for specific node to be turned off.

$$\text{batchnorm}(x) = \frac{x - \text{mean}_{\text{batch}}(x)}{\sqrt{\text{var}_{\text{batch}}(x) + \epsilon}} \gamma + \beta, \quad (3.5.4)$$

where  $\text{mean}_{\text{batch}}(x)$  and  $\text{var}_{\text{batch}}(x)$  calculate the mean and variance over a batch, respectively. Further,  $\gamma$  and  $\beta$  are trainable layer parameters and  $\epsilon$  is a numerical factor to prevent division by 0.

By applying a batchnorm layer to the output of every network layer one can ensure that the inputs to all intermediate layers are also normalized. The exact means and variances are constantly recalculated using the current batch. During evaluation, this would make the exact network output for a given data point depend on the makeup of the batch it is in. To prevent such behavior batchnorm layers keep a running average over the mean and variance during training and use this average during evaluation.

### Layer Normalization Layers

An alternative approach to intra-network normalization is Layer Normalization (layernorm). Its operating principle is similar to that of batchnorm, however, it uses the transformation

$$\text{layernorm}(x) = \frac{x - \text{mean}_{\text{input}}(x)}{\sqrt{\text{var}_{\text{input}}(x) + \epsilon}} \gamma + \beta. \quad (3.5.5)$$

This is identical to the batchnorm transformation except it uses the mean and variance calculated over parts of the input  $\text{mean}_{\text{input}}(x)$  and  $\text{var}_{\text{layer}}(x)$ , instead of the mean and variance over the batch. Therefore layernorm is independent of the batch makeup and does not require special treatment during evaluation. However, this comes at the cost of normalizing several input features at once, rather than independently, as is the case in batchnorm.

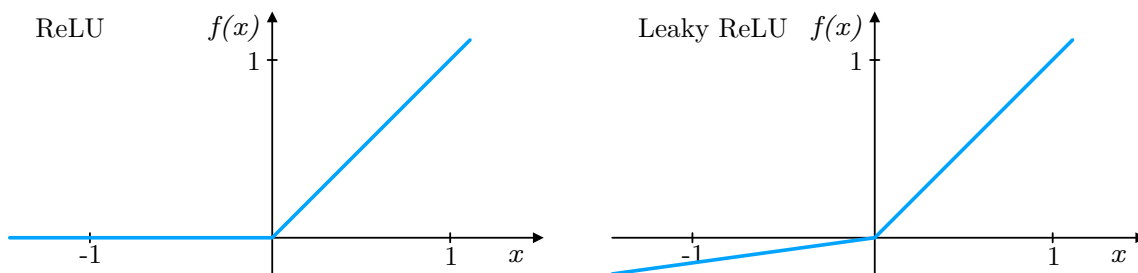


Figure 3.8: Depiction of ReLU and Leaky ReLU activation functions.

### 3.6 Activation Functions

The basic neural network layers, such as dense layers or convolutional layers are linear operations, expressible as matrix multiplications. Therefore any network built as a combination of these layers is itself again a linear function. However, many of the problems and functions one aims to describe with a network are inherently nonlinear, and therefore difficult to approximate with a purely linear function.

Activation functions were introduced to address this problem by adding nonlinearity to the neural network. These activation functions are nonlinear mappings applied to the output of a network layer before it is passed to the next layer.

There exists a wide range of activation functions, many of which with very specialized applications, therefore this section will only focus on those functions relevant to this thesis.

#### ReLU

One of the simplest, yet most effective activation functions, is the Rectified Linear Unit (ReLU) [113]. The function is shown in the left panel of Figure 3.8 and corresponds to an identity mapping for inputs larger than zero, and a mapping to zero for input values below zero. It can be defined as

$$f_{\text{ReLU}}(x) = \max(x, 0). \quad (3.6.1)$$

ReLU provides a straightforward way of introducing nonlinearity to a model without requiring any complex computations. This ease of use and fast evaluation has made the function a mainstay in modern ML. However, ReLU has a notable downside, since its gradient for inputs below zero is once again zero, leading to the aforementioned vanishing gradient problem.

#### Leaky-ReLU

Leaky ReLU [114] is a modification of ReLU that aims to address the vanishing gradient problem. To this end, leaky ReLU introduces a small slope in the negative input range, indicated in the right panel of Figure 3.8. The leaky ReLU function is given by

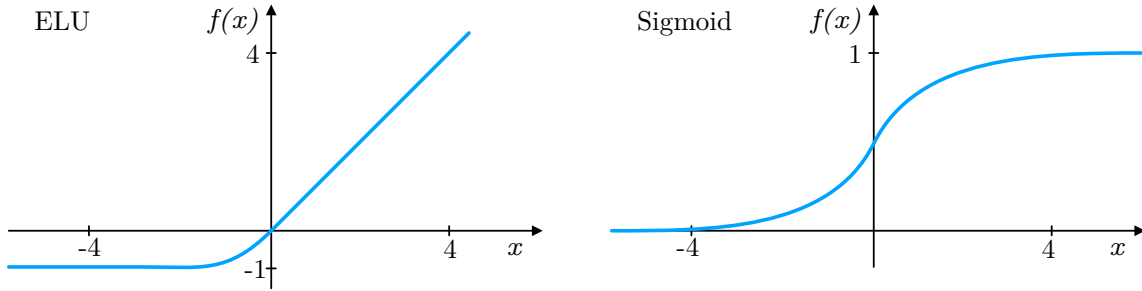


Figure 3.9: Depiction of ELU and Sigmoid activation functions.

$$f_{\text{ReLU}}(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}, \quad (3.6.2)$$

where  $\alpha$  is a parameter dictating the negative slope. The introduction of this negative slope means the gradient of  $f_{\text{ReLU}}$  can no longer become zero in the negative region, thereby reducing the risk of vanishing gradients.

### ELU

An Exponential Linear Unit (ELU) [115] can be used in cases where the sharp cutoffs of either ReLU or leaky ReLU are not desired. The ELU function extends the ReLU function into the negative input range by an exponential function, defined as

$$f_{\text{ReLU}}(x) = \begin{cases} x, & x \geq 0 \\ \alpha(\exp(x) - 1), & x < 0 \end{cases}, \quad (3.6.3)$$

where  $\alpha$  is once again a scalable parameter. This results in a smooth transition from identity mapping to a constant value, as can be seen on the left panel of Figure 3.9.

### Sigmoid

Certain applications, such as classification tasks, require the network outputs to be constrained between 0 and 1. To map the, in principle, unconstrained network output to this interval one can use the Sigmoid function, defined as

$$f_{\text{Sigmoid}}(x) = \frac{1}{1 + \exp(-x)}. \quad (3.6.4)$$

The right panel of Figure 3.9 shows this function. One can see the asymptotic behavior of  $\lim_{x \rightarrow -\infty} f_{\text{Sigmoid}}(x) = 0$  and  $\lim_{x \rightarrow \infty} f_{\text{Sigmoid}}(x) = 1$  that ensures the final output is constrained to the desired range of  $[0, 1]$  while maintaining differentiability.



## Chapter 4

# Generative Models

Generative models describe a subset of ML models designed to learn an underlying distribution from a given sample of training data and then produce new samples from this distribution. The underlying principle of all generative models covered in this work is to have a neural network map an input consisting of random noise onto new samples that lie in the target distribution. Defining the architecture of this model is largely dependent on the data set one intends to generate new samples for and optimizing this can be a time-intensive process. The main difficulty in utilizing generative models however is training them. Unlike in the classification examples discussed in Section 3.3, defining a training objective for generative models is highly non-trivial. This objective needs to simultaneously measure how well the samples produced fit the training distribution, while still being differentiable. The main differences between the models described in the following relate to how they approach this problem.

In this chapter, we will initially discuss the challenges of building generative models in Section 4.1. Following this, individual generative methods will be explored, starting with Generative Adversarial Networks in Section 4.2 and Wasserstein generative adversarial networks in Section 4.3. We then move to Autoencoder-based generative models, starting with an introduction to Autoencoders in Section 4.4. Based on this, Variational and Adversarial Autoencoders are covered in Sections 4.5 and 4.6, respectively. Section 4.7 introduces the Bounded Information Bottleneck Autoencoder, an overarching model comprising several other generative approaches. Normalizing Flow architectures are described in Section 4.8, and Section 4.9 introduces the concept of generative models with conditional input, as well as how to modify the previously covered generative models into conditional models.

### 4.1 Challenges of Generative Models

Generative models have their own set of potential failure cases, that are instructive to briefly highlight before exploring the methodology of building generative approaches. To illustrate the difficulties, the MNIST handwritten number data set [116] will be used. This set contains images of written single-digit numbers ranging from 0 to 9 and is often used to benchmark ML approaches.

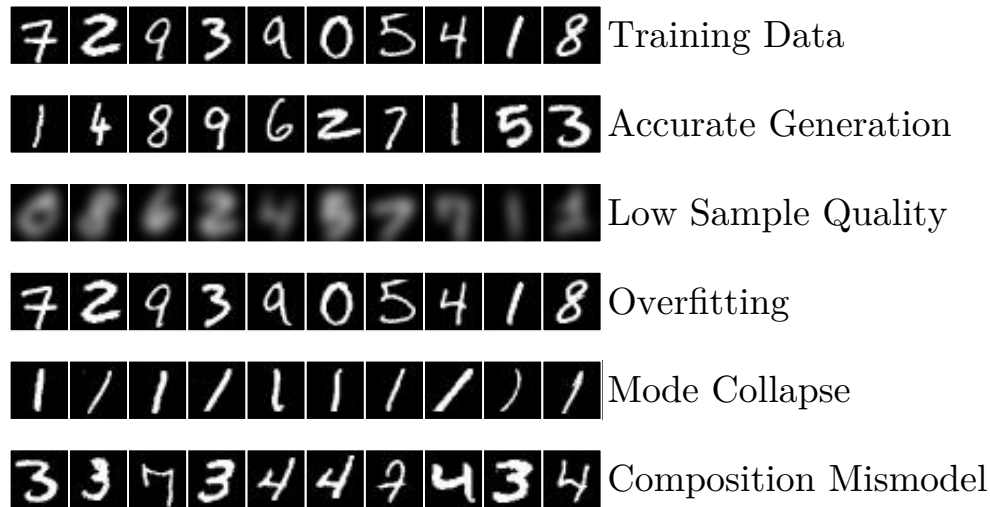


Figure 4.1: Examples of generative ML model failure cases using the MNIST handwritten data set as an example. Specific explanations are outlined in the text.

### Low Sample Quality

The main goal of a generative model is to produce new samples that lie within the underlying distribution of the training data. One common failure case is that the samples generated by the model deviate from that underlying distribution. The extent of this can vary, ranging from the model producing random noise that is in no way comparable to the training data, to more subtle differences in the generated data. In terms of the distributions, this means the generated images are from a, for example, smeared-out distribution.

Low-quality samples are a generally easy-to-spot problem, as can be seen through direct comparison of the generated and training data.

In the MNIST example shown in Figure 4.1, the low sample quality case is demonstrated by several images that do not represent handwritten numbers.

### Overfitting

Generative overfitting is similar to overfitting in general ML and describes a case where the model learns the training data by heart. This results in a generative model that produces exact copies of the training data. This is a significant problem as it means the generative model does not produce new data points, effectively defeating its purpose.

Overfitting is comparatively rare in generative models, especially for highly complex data sets, nevertheless, it can be a problem. Spotting overfitting in a generative model often requires extensive comparisons of generated samples and training data to ensure there are no direct matches.

## Mode Collapse

Mode collapse occurs when a generative model trained on a data set with multiple classes fails to reproduce all the possible classes. For example, a mode collapsed model trained on the MNIST handwritten numbers may only produce samples showing the number 1, as indicated in Figure 4.1. In terms of distributions, this can be understood as the generative model learning only one mode of a multimodal distribution. Importantly mode collapse does not mean the individual generated samples are of low quality, only that some classes are never reproduced.

## Data Composition Mismatching

In some cases, a generative model may produce individual, high-quality samples, but the overall distribution of the generated samples does not match that of the training data. In the MNIST data set this could, for example, correspond to certain numbers being more abundant in the generated data.

There are several generative applications where the overall data composition is not relevant. One such example is generative art [117], where great importance is placed on the quality of individual images, but global distributions of properties of the produced artworks are of no consequence.

The situation is different when one intends to use generative models for fast simulation in HEP. Since particle physics almost exclusively operates on the distribution level, it is vitally important to replicate the overall properties of the data one aims to generate. This presents an additional challenge when applying generative models in particle physics, which is not present in many computer science applications.

## 4.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [118] are a common approach to generative modeling. The core of a GAN consists of a so-called generator network  $G$  with parameters  $\theta_g$ . This network takes random noise variables  $z$ , drawn from a distribution  $p_z(z)$ , as its input and maps these onto the generator dataspace  $p_g$ , with this mapping expressed as  $G(z, \theta_g)$ . In order to train this generator one employs a second classification network, called a discriminator  $D$ . This discriminator is tasked with distinguishing between the fake samples produced by the generator  $G(z, \theta_g)$  and real samples from the training set  $x$ . In other words,  $D$  is trained to maximize the cross entropy term,

$$\max_D \log(D(x, \theta_d)) + \log(1 - D(G(z, \theta_g), \theta_d)) . \quad (4.2.1)$$

The output of the discriminator is then used to train the generator by having the generator maximize the confusion of the discriminator

$$\min_G \log(1 - D(G(z, \theta_g), \theta_d)) . \quad (4.2.2)$$

In combination this means the discriminator and generator try to either maximize or minimize the GAN loss function given by:

$$\min_G \max_D \mathbb{E}_x[\log(D(x, \theta_d))] + \mathbb{E}_z[\log(1 - D(G(z, \theta_g), \theta_d))] , \quad (4.2.3)$$

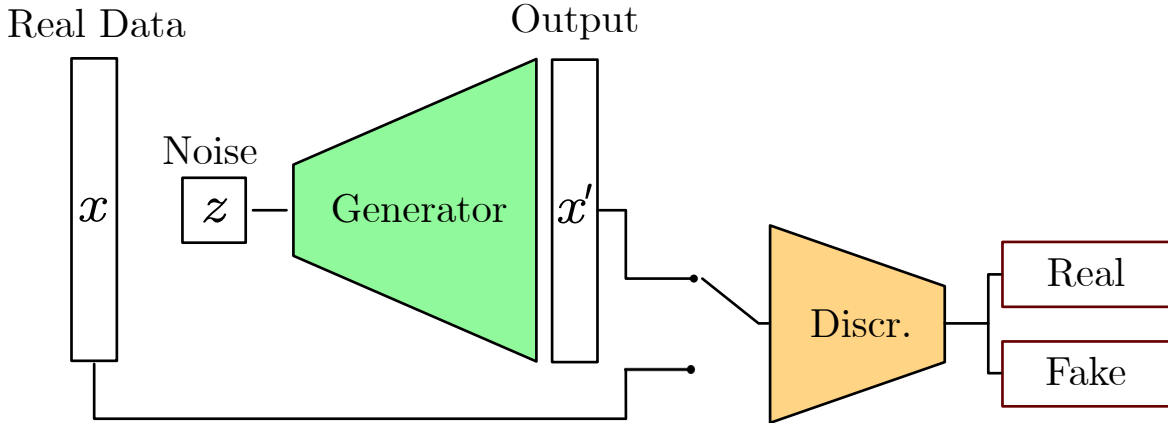


Figure 4.2: Schematic depiction of a GAN setup. The green model shows the generator  $G$ , while the orange model is the discriminator  $D$ . For brevity the generator output  $G(z)$  is labeled  $x'$ .

where  $\mathbb{E}_x$  and  $\mathbb{E}_z$  are the expectation values over the data  $x$  and the random noise  $z$  respectively. This training objective can be understood as a back and forth between the generator and discriminator, where the discriminator learns to identify errors made by the generator, and the discriminator output is then used to train the generator to fix these errors. Figure 4.2 shows the GAN training scheme. The GAN approach solves the aforementioned difficulty of defining a generative loss function by deputizing the discriminator to act as this loss function. This adversarial training can be shown to minimize the Jensen-Shannon divergence between the training distribution and the generator distribution [118]. The Jensen-Shannon divergence is a symmetric measure of the similarities between two distributions. Therefore by minimizing this divergence between training and generated data, the generator learns to approximate the real data.

In an idealized scenario both the generator and discriminator improve continuously until they reach a Nash equilibrium, where the discriminator can no longer distinguish between the generator-produced samples and the training samples. In practice training a GAN to convergence can be challenging, as the adversarial training has inherent instabilities. Additionally, one runs the risk of designing either the generator or discriminator to be too powerful. This leads to a regime where the discriminator cannot provide useful gradients to the generator, causing the GAN training to stall.

### 4.3 Wasserstein GANs

The Wasserstein GAN (WGAN) [119] is a modification of the standard GAN that aims to improve some of the shortcomings of the GAN setup. While a GAN aims to minimize the Jensen-Shannon divergence between the generator and training distributions, a WGAN attempts to minimize the Wasserstein distance between the two distributions. The Wasserstein-1 (W-1) distance, also referred to as the Earth Mover distance originates, from optimal transport theory, and describes the “work” required to move the mass of one distribution to another. In the WGAN case with the training distribution  $p_x$  and the generator distribution  $p_g$ , the W-1 distance can be expressed as

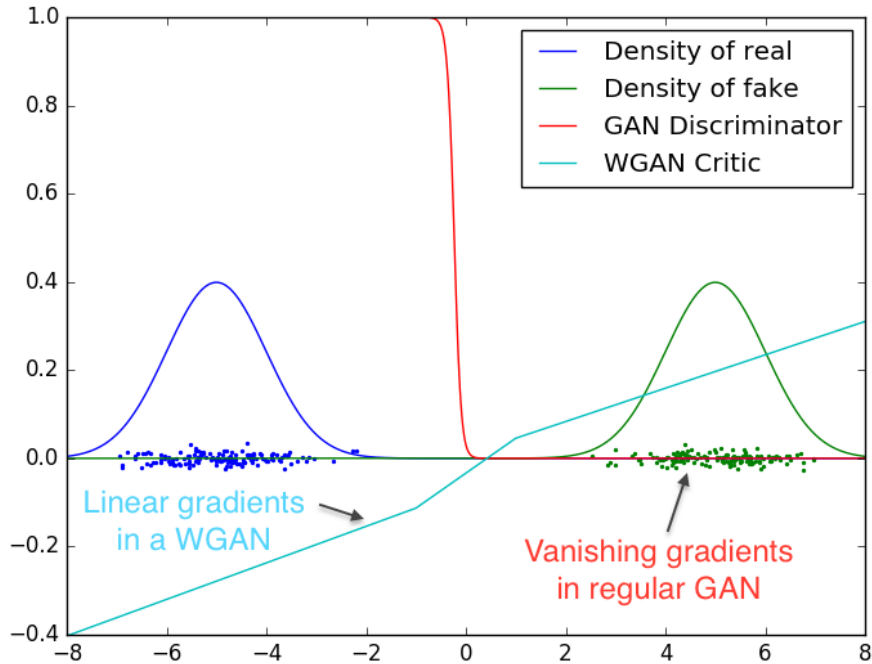


Figure 4.3: Adversarial network outputs for input with a value shown on the x-axis. In this case, the training and generator density are disjoint. The GAN discriminator output approximating the Jensen-Shannon divergence can be observed to have a vanishing gradient anywhere except around 0. The WGAN critic approximating the W-1 distance maintains a clear gradient across the entire range. Image taken from [119].

$$W(p_x, p_g) = \inf_{\gamma \in \Pi(p_x, p_g)} \mathbb{E}_{(a,b) \sim \gamma} [|a - b|]. \quad (4.3.1)$$

Here  $\Pi(p_x, p_g)$  can be understood as the space of all transport plans that map from  $p_x$  to  $p_g$ .

The W-1 distance offers a significant advantage compared to the Jensen-Shannon divergence in cases where  $p_x$  and  $p_g$  do not overlap. Figure 4.3 demonstrates an example where  $p_x$  and  $p_g$  are disjoint, as will commonly happen close to the start of the training. In this case, the Jensen-Shannon divergence has a vanishing gradient, potentially causing the GAN training to get stuck, whereas the W-1 distance has a clear gradient.

One difficulty with using the W-1 distance, however, is that the expression in Equation (4.3.1) is extremely difficult to compute for higher dimensions. However, one can rewrite the W-1 distance into its dual form using the Kantorovich-Rubinstein duality [120]:

$$W(p_x, p_g) = \sup_{\|f\|_{L \leq 1}} \mathbb{E}_{a \sim p_x} [f(a)] - \mathbb{E}_{b \sim p_g} [f(b)], \quad (4.3.2)$$

where  $\sup_{\|f\|_{L \leq 1}}$  is the supremum over all 1-Lipschitz functions, meaning all functions for which

$$|f(x_1) - f(x_2)| < k|x_1 - x_2| \quad \text{with } k = 1, \quad (4.3.3)$$

holds true. We can extend Equation (4.3.2) from 1-Lipschitz functions to k-Lipschitz functions by scaling the distance  $W(p_x, p_g)$  by k, resulting in

$$k W(p_x, p_g) = \sup_{\|f\|_{L \leq k}} \mathbb{E}_{a \sim p_x}[f(a)] - \mathbb{E}_{a \sim p_g}[f(a)]. \quad (4.3.4)$$

This multiplicative factor can be disregarded when using the W-1 distance as a loss function, as it has no impact on the direction of the resulting gradients. Therefore we can use a function expressed by a neural network to estimate the W-1 distance, as long as this function is a k-Lipschitz function. The condition for a k-Lipschitz function in Equation (4.3.3) is equivalent to requiring the function to be differentiable almost everywhere and to have a bounded derivative  $\frac{df}{dx} < k$ . The differentiability condition will be inherently fulfilled by a given neural network, as they are by construction differentiable. Furthermore, any network with weights  $\theta$  that lie within a bounded space will have a bounded derivative<sup>1</sup>. Therefore one can ensure a network is a k-Lipschitz function by limiting its weights to an arbitrary maximum value. This approach, known as weight clipping, was used in the original WGAN implementation [119]. A network constrained in this manner can then be used to estimate the Wasserstein distance between the generated and real distributions. For an adversarial network  $D$  with parameters  $\theta_d$  this estimation is given by

$$W(p_x, p_g) \approx \mathbb{E}_{a \sim p_x}[D(a, \theta_d)] - \mathbb{E}_{a \sim p_g}[D(a, \theta_d)]. \quad (4.3.5)$$

Using this one can write the discriminator loss function of the WGAN with a generator  $G$  with parameters  $\theta_g$  as

$$L_{\text{disc}} = \mathbb{E}_{z \sim p_z}[D(G(z, \theta_g), \theta_d)] - \mathbb{E}_{a \sim p_x}[D(a, \theta_d)], \quad (4.3.6)$$

and the generator loss as

$$L_{\text{gen}} = -\mathbb{E}_{z \sim p_z}[D(G(z, \theta_g), \theta_d)] \quad (4.3.7)$$

One notable feature of this loss function compared to Equation (4.2.2) is the lack of cross entropy-induced logarithms. This means the discriminator network is no longer constrained to an output between  $[0, 1]$ . An intuitive way of understanding this is that the adversarial network in a WGAN setup does not try to determine whether a given input is either real or fake, but rather to rate the “realness” of an input using an arbitrary score scale. For this reason, the adversary in a WGAN is commonly called a *critic* rather than a *discriminator*. Figure 4.4 shows a WGAN setup, note the change to a critic network that returns a score, compared to the discriminator in the GAN setup.

---

<sup>1</sup>Assuming no non-standard activation functions with an inherently unbounded derivative, such as exponential functions, are used.

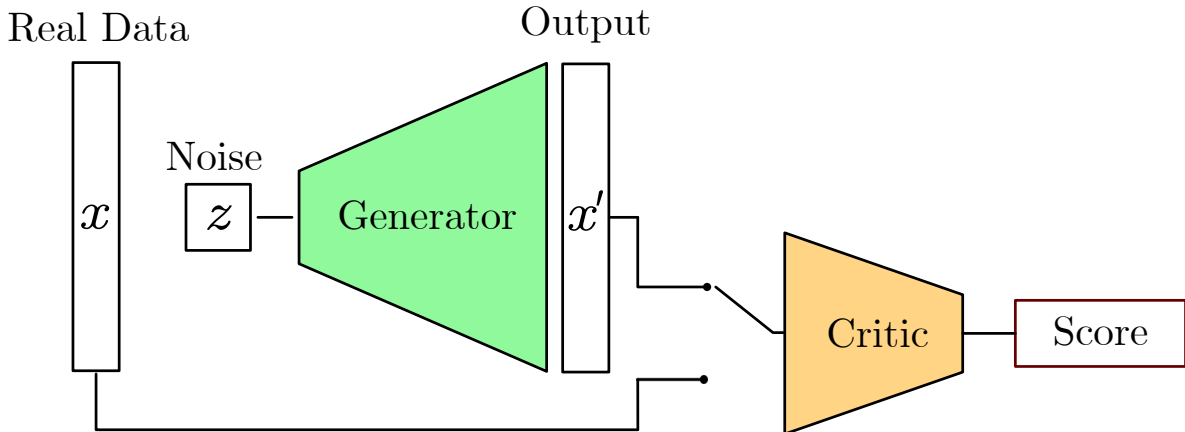


Figure 4.4: Schematic depiction of a WGAN setup. The green model shows the generator  $G$ , the orange model is the critic  $D$ . For brevity the generator output  $G(z)$  is labeled  $x'$ .

The weight-clipping approach has, as the original WGAN paper points out, several shortcomings, such as long convergence times if a large clipping maximum is chosen and vanishing gradients in cases where the maximum is too small.

An improved approach to WGAN training [121] makes use of a gradient penalty term to enforce the  $k$ -Lipschitz constraint. Here, one first defines a new distribution  $p_s$ , that consists of points obtained by uniformly interpolating between pairs of points from the real distribution  $p_x$  and the generator distribution  $p_g$ . One then calculates the gradient  $\nabla_s D(s)$  of the critic network  $D$  with regard to a sample  $s$  drawn from the interpolation distribution  $p_s$ . This gradient can be understood to approximate the gradient of  $D$  in a region between the real distribution and the generator distribution. To ensure the  $k$ -Lipschitz condition, this gradient needs to be constrained. This is achieved by taking the squared difference between the absolute of the gradient and 1 so that the resulting expression is minimal if the gradient of the adversarial network is 1. This then leads to the full gradient penalty critic WGAN loss of

$$L_{\text{disc}} = \mathbb{E}_{z \sim p_z} [D(G(z, \theta_g), \theta_d)] - \mathbb{E}_{a \sim p_x} [D(a, \theta_d)] + \mathbb{E}_{s \sim p_s} [(|\nabla_s D(s, \theta_d)| - 1)^2]. \quad (4.3.8)$$

For the remainder of this work, the gradient penalty trained WGAN will be assumed as the default WGAN setup.

## 4.4 Autoencoders

AutoEncoders (AEs) [106] are most commonly found in unsupervised ML applications such as data compression [122–124] and anomaly detection [125, 126]. They are not inherently generative models, however, they form the foundation of a variety of generative models, such as the Variational Autoencoder, Adversarial Autoencoder, and Bounded Information-Bottleneck Autoencoder discussed in the subsequent chapters. Therefore, their general principles will be briefly introduced here.

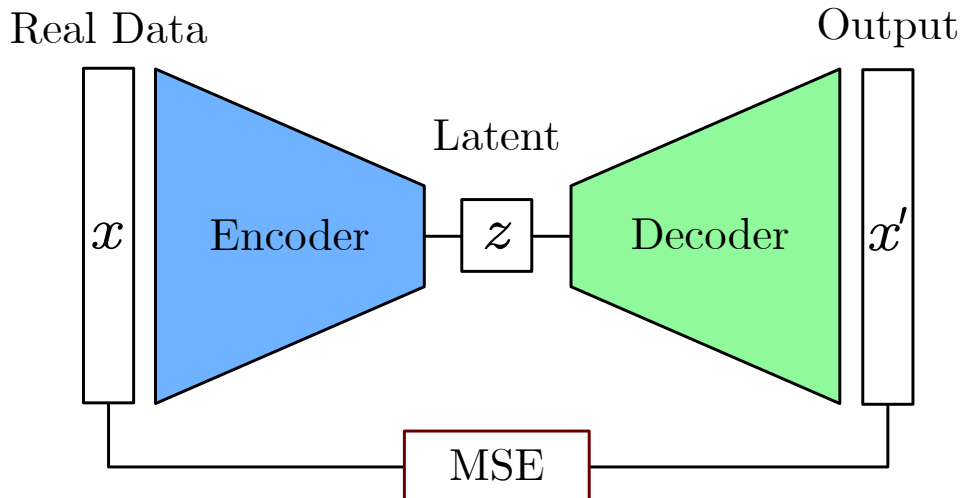


Figure 4.5: Schematic depiction of an AE setup. The green model shows the encoder  $E$ , the blue model is the decoder  $D$ ,  $x'$  is the output of the decoder.

Fundamentally, AEs consist of two parts, one so-called encoder network  $E$  with parameters  $\theta_e$  and one decoder network  $D$  with parameters  $\theta_d$ . The encoder maps samples  $x$  from the distribution of real data  $p_x$  onto samples  $z$  from a lower dimensional latent space  $p_z$  and the decoder maps the samples from this latent space back to the data space. A schematic view of this mapping is shown in Figure 4.5. The latent space can be understood as an  $n$ -dimensional vector, where  $n$  is a tunable parameter called the latent space size. The latent space is also often referred to as a bottleneck, in cases where  $n$  is smaller than the input dimensionality.

In order to train an AE one passes the training data sequentially through the encoder and decoder network with a loss function consisting of the squared element-wise difference between the input that was given to the encoder and the output of the decoder. This can be expressed as

$$L_{\text{AE}} = \mathbb{E}_{x \sim p_x} [(x - D(E(x, \theta_e), \theta_d))^2]. \quad (4.4.1)$$

This means the main training objective of an AE is to reconstruct the original input images as closely as possible. Ideally, an AE would effectively be the identity operation. However, since the latent space between the encoder and decoder has a lower dimension than the input and output, a lossless mapping from input to output is often not possible. Therefore, the AE is forced to find the most efficient representation of the input data, where only the features that are required for the reconstruction of the input are preserved in the latent space. The compressed representation will often rely on features specific to the data set that was used to train the AE. In an ideal case, this leads to the deviation between the original input and the reconstructed output being small for data similar to the training data, and large for data not present during training. Therefore, this reconstruction-deviation can be used to identify outliers or anomalous points in a data set.

However, it needs to be noted that this behavior is not universally guaranteed. For example, in cases where the anomalous data is less complex than the bulk of the training data the reconstruction deviation can actually be smaller for the anomalous points [127].



## 4.5 Variational Autoencoders

The AE structure is of interest for generative applications as the decoder part of the AE already fulfills a role similar to a generative model, in that it maps a low dimensional latent space onto the data space. Furthermore, the AE setup circumvents the difficulties of defining a generative loss function by directly comparing individual training data points with individual generated (reconstructed) points. Therefore, the problem of deciding if the output is realistic is never raised, as one is only concerned with whether or not the input and output look alike.

Therefore, if one could generate a new sample from the latent space of an AE and then pass this sample through the decoder, one could use this to generate new data similar to the training data. However, the latent space is entirely unconstrained. This makes sampling from the latent space a nearly impossible task.

Variational Autoencoders (VAEs) [128] aim to address this difficulty by regularizing the latent space. In order to further understand how this regularization works we need to view the VAE from a probability model perspective. The following derivation follows Reference [129]. From a probabilistic perspective, the model can be seen to have learned the joint probability

$$p(x, z) = p(x|z)p(z) , \quad (4.5.1)$$

where  $p(z)$  is the prior distribution of the latent space, and  $p(x|z)$  is the probability of generating a sample  $x$  for a given latent sample  $z$ .  $p(x|z)$  can be understood as the decoder network, mapping from an arbitrary latent space to the data space. The next goal is now to find the encoder, defined by  $p(z|x)$ . Using Bayes Theorem this can be rewritten as

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} . \quad (4.5.2)$$

The denominator is given by the integral

$$p(x) = \int p(x|z)p(z)dz , \quad (4.5.3)$$

which is not tractable. However, it can be estimated using variational inference, the namesake of the VAE. For this purpose, one defines an approximation distribution  $q(z|x)$ , chosen from a group of distributions that are inherently tractable, such as Gaussians. Then, one manipulates  $q(z|x)$  in a way that makes it very similar to the target distribution  $p(z|x)$ , by varying the parameters to minimize the Kullback–Leibler divergence between the two,  $\text{KL}[q(z|x), p(z|x)]$ . This can be expressed as

$$\text{KL}[q(z|x), p(z|x)] = - \sum_z q(z|x) \log \left( \frac{p(z|x)}{q(z|x)} \right) . \quad (4.5.4)$$

We can once again rewrite this using Equation (4.5.2) as

$$\text{KL}[q(z|x), p(z|x)] = - \sum_z q(z|x) \log \left( \frac{p(x|z)p(z)}{q(z|x)} \right) \quad (4.5.5)$$

$$\text{KL}[q(z|x), p(z|x)] = - \sum_z q(z|x) \log \left( \frac{p(x|z)p(z)}{q(z|x)} \frac{1}{p(x)} \right) \quad (4.5.6)$$

$$\text{KL}[q(z|x), p(z|x)] = - \sum_z q(z|x) \left[ \log \left( \frac{p(x|z)p(z)}{q(z|x)} \right) - \log(p(x)) \right]. \quad (4.5.7)$$

Using Equation (4.5.1) we can express  $p(x|z)p(z)$  as  $p(x, z)$

$$\text{KL}[q(z|x), p(z|x)] = - \sum_z q(z|x) \left[ \log \left( \frac{p(x, z)}{q(z|x)} \right) - \log(p(x)) \right] \quad (4.5.8)$$

$$\text{KL}[q(z|x), p(z|x)] = - \sum_z q(z|x) \log \left( \frac{p(x, z)}{q(z|x)} \right) + \sum_z q(z|x) \log(p(x)) \quad (4.5.9)$$

Since  $\log(p(x))$  does not depend on  $z$  and the sum over  $q(z|x)$  for all  $z$  and a given  $x$  is 1, one can write  $\sum_z q(z|x) \log(p(x))$  as  $\log(p(x))$

$$\text{KL}[q(z|x), p(z|x)] = - \sum_z q(z|x) \log \left( \frac{p(x, z)}{q(z|x)} \right) + \log(p(x)). \quad (4.5.10)$$

This expression can be reordered as

$$\log(p(x)) = \text{KL}[q(z|x), p(z|x)] + \sum_z q(z|x) \log \left( \frac{p(x, z)}{q(z|x)} \right). \quad (4.5.11)$$

Since we are varying  $q(z|x)$ , we can assume  $p(x)$  to be constant for any given  $x$ . Therefore, minimizing the KL-Divergence  $\text{KL}[q(z|x), p(z|x)]$  is equivalent to maximizing the last expression in Equation (4.5.11),  $\sum_z q(z|x) \log \left( \frac{p(x, z)}{q(z|x)} \right)$ , as both terms always sum up to a constant. This expression is also known as the Evidence Lower Bound (ELBO), it can be further rewritten as

$$\text{ELBO} = \sum_z q(z|x) \log \left( \frac{p(x, z)}{q(z|x)} \right) \quad (4.5.12)$$

$$\text{ELBO} = \sum_z q(z|x) \log \left( \frac{p(x|z)p(z)}{q(z|x)} \right) \quad (4.5.13)$$

$$\text{ELBO} = \sum_z q(z|x) \left[ \log(p(x|z)) + \log \left( \frac{p(z)}{q(z|x)} \right) \right]. \quad (4.5.14)$$

$$\text{ELBO} = \sum_z q(z|x) \log(p(x|z)) + \sum_z q(z|x) \log \left( \frac{p(z)}{q(z|x)} \right). \quad (4.5.15)$$

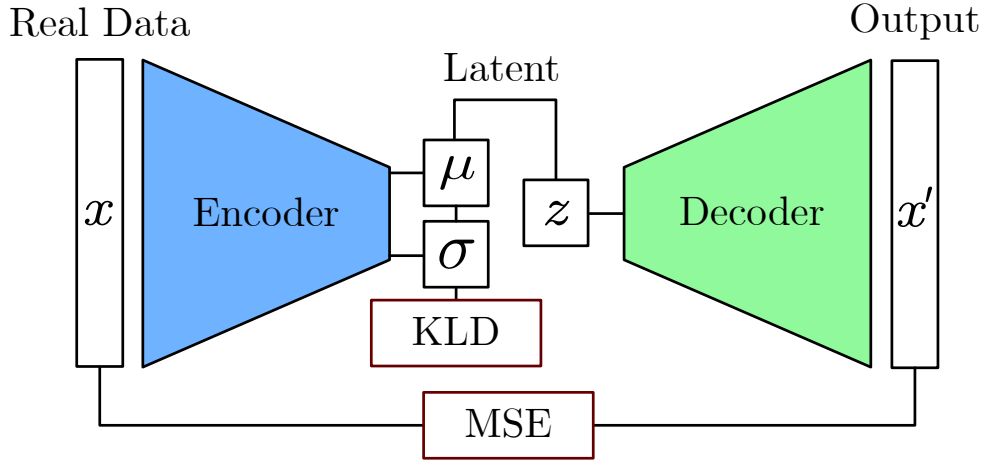


Figure 4.6: Schematic depiction of a VAE setup. The green model shows the encoder  $E$ , the blue model is the decoder  $D$ , and  $x'$  is the output of the decoder. The latent space sample  $z$  is obtained by sampling from the Gaussian distribution defined by  $\mu$  and  $\sigma$ .

The first part of this equation can be seen as the expectation of  $\log(p(x|z))$  for a given  $q(z|x)$ . Furthermore, comparing the second part of this expression with Equation (4.5.4) reveals that this is equal to the negative KL-Divergence between  $p(z)$  and  $q(z|x)$ , resulting in

$$\text{ELBO} = \mathbb{E}_{q(z|x)} \log(p(x|z)) - \text{KL}[q(z|x), p(z)] . \quad (4.5.16)$$

This means that, in order to maximize the ELBO, the VAE needs to maximize the expected logarithm of  $p(x|z)$ , while minimizing the KL-Divergence between the output of the encoder  $q(z|x)$  and the given latent space  $p(z)$ . The first objective is generally achieved through the use of an MSE-like reconstruction loss, similar to the standard AE while calculating the KL-Divergence will depend on the specific chosen form of  $p(z)$  and  $q(z|x)$ . One requirement is that  $q(z|x)$  needs to be tractable. The common solution [128] to this is to restrict the latent space  $q(z|x)$  to a multidimensional uncorrelated Gaussian distribution. In practice, this is achieved by having the encoding network output two  $n$ -dimensional vectors, where  $n$  is the size of the latent space. This two-vector latent space is shown in the VAE architecture depicted in Figure 4.6. The first vector is then treated as the means  $\mu$  of the Gaussians, and the second is treated as the diagonal variances of the Gaussians  $\sigma$ . The individual latent space samples are then obtained by sampling from these Gaussian distributions. In practice, this sampling is non-trivial, as one needs to ensure it is still possible to calculate the network gradients despite the sampling operation, in order to train the encoder. To this end, one employs the reparametrization trick: first one draws a sample vector  $s$  from an  $n$ -dimensional Gaussian  $\mathcal{N}(0, 1)$  with means 0 and widths 1. Then  $s$  is shifted by  $\mu$  and scaled by  $\sigma$ ,

$$\hat{s} = \sigma s + \mu . \quad (4.5.17)$$

The resulting  $\hat{s}$  has the same properties as if drawn directly from  $\mathcal{N}(\mu, \sigma)$ , but is directly linked to  $\mu$  and  $\sigma$  via addition and multiplication, allowing gradients to propagate through the sampling operation.

As we want sampling from the target latent space  $p(z)$  to be easy, we choose  $p(z)$  to be a multidimensional Gaussian with  $\mu = 0$  and  $\sigma = 1$ . This modifies the KL-Divergence term in Equation (4.5.16) to

$$\text{KL}[q(z|x), p(z)] = \text{KL}[\mathcal{N}(\mu, \sigma), \mathcal{N}(0, 1)] . \quad (4.5.18)$$

Since both  $\mathcal{N}(\mu, \sigma)$  and  $\mathcal{N}(0, 1)$  have diagonal covariance matrices, we can reduce the multidimensional KL-divergence to a sum over the individual dimensions,

$$\text{KL}[\mathcal{N}(\mu, \sigma), \mathcal{N}(0, 1)] = \sum_{i=0}^n \text{KL}[\mathcal{N}(\mu_i, \sigma_i), \mathcal{N}(0, 1)] . \quad (4.5.19)$$

Using the continuous definition of the KL-divergence,

$$\text{KL}[q(x), p(x)] = - \int_x q(x) \log\left(\frac{p(x)}{q(x)}\right) dx , \quad (4.5.20)$$

the KL-divergence between two Gaussians can be further transformed. For the sake of readability we define  $\mathcal{N}_1(x) = \mathcal{N}(\mu_i, \sigma_i)(x)$  and  $\mathcal{N}_2(x) = \mathcal{N}(0, 1)(x)$

$$\text{KL}[\mathcal{N}_1, \mathcal{N}_2] = - \int_x \mathcal{N}_1(x) \log\left(\frac{\mathcal{N}_2(x)}{\mathcal{N}_1(x)}\right) dx \quad (4.5.21)$$

$$\text{KL}[\mathcal{N}_1, \mathcal{N}_2] = - \int_x \mathcal{N}_1(x) \log(\mathcal{N}_2(x)) dx + \int_x \mathcal{N}_1(x) \log(\mathcal{N}_1(x)) dx \quad (4.5.22)$$

The second integral is the negative entropy of a normal distribution and can be solved as

$$\int_x \mathcal{N}_1(x) \log(\mathcal{N}_1(x)) dx = -\frac{1}{2} - \frac{1}{2} \log(2\pi\sigma_i^2) . \quad (4.5.23)$$

The first integral is more complex, however we can leverage the fact that  $\mathcal{N}_2$  is a normal distribution and insert  $\mathcal{N}_2(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$ , resulting in

$$\int_x \mathcal{N}_1(x) \log(\mathcal{N}_2(x)) dx = \int_x \mathcal{N}_1(x) \log\left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)\right) dx \quad (4.5.24)$$

$$\int_x \mathcal{N}_1(x) \log(\mathcal{N}_2(x)) dx = -\frac{1}{2} \log(2\pi) + \int_x \mathcal{N}_1(x) \log\left(\exp\left(-\frac{x^2}{2}\right)\right) dx . \quad (4.5.25)$$

$$\int_x \mathcal{N}_1(x) \log(\mathcal{N}_2(x)) dx = -\frac{1}{2} \log(2\pi) - \int_x \mathcal{N}_1(x) \frac{x^2}{2} dx . \quad (4.5.26)$$

The integral  $\int_x \mathcal{N}_1(x) \frac{x^2}{2}$  is equal to the expectation value of  $x^2$ ,  $\mathbb{E}_{x \sim \mathcal{N}_1}(x^2)$ . Using the definition of the variance  $\text{var}_{x \sim \mathcal{N}_1}(x) = \mathbb{E}_{x \sim \mathcal{N}_1}(x^2) - \mathbb{E}_{x \sim \mathcal{N}_1}(x)^2$  one can express this as

$$\int_x \mathcal{N}_1(x) \log(\mathcal{N}_2(x)) dx = -\frac{1}{2} \log(2\pi) - \frac{1}{2} (\mathbb{E}_{x \sim \mathcal{N}_1}(x^2)) \quad (4.5.27)$$

$$\int_x \mathcal{N}_1(x) \log(\mathcal{N}_2(x)) dx = -\frac{1}{2} \log(2\pi) - \frac{1}{2} \left( \text{var}_{x \sim \mathcal{N}_1}(x) + \mathbb{E}_{x \sim \mathcal{N}_1}(x^2) \right). \quad (4.5.28)$$

Since  $\mathcal{N}_1$  is a Gaussian with mean  $\mu_i$  and width  $\sigma_i$ , the distributions expectation value and variance are given by  $\mathbb{E}_{x \sim \mathcal{N}_1}(x)^2 = \mu_i^2$  and  $\text{var}_{x \sim \mathcal{N}_1}(x) = \sigma_i^2$ . Therefore the integral becomes

$$\int_x \mathcal{N}_1(x) \log(\mathcal{N}_2(x)) dx = -\frac{1}{2} \log(2\pi) - \frac{1}{2} (\sigma_i^2 + \mu_i^2). \quad (4.5.29)$$

Using this we can write the full KL-divergence as

$$\text{KL}[\mathcal{N}_1, \mathcal{N}_2] = - \int_x \mathcal{N}_1(x) \log(\mathcal{N}_2(x)) dx + \int_x \mathcal{N}_1(x) \log(\mathcal{N}_1(x)) dx \quad (4.5.30)$$

$$\text{KL}[\mathcal{N}_1, \mathcal{N}_2] = - \left( -\frac{1}{2} \log(2\pi) - \frac{1}{2} (\sigma_i^2 + \mu_i^2) \right) + \left( -\frac{1}{2} - \frac{1}{2} \log(2\pi\sigma_i^2) \right) \quad (4.5.31)$$

$$\text{KL}[\mathcal{N}_1, \mathcal{N}_2] = -\frac{1}{2} \log(\sigma_i^2) + \frac{1}{2} (\sigma_i^2 + \mu_i^2) - \frac{1}{2} \quad (4.5.32)$$

$$\text{KL}[\mathcal{N}_1, \mathcal{N}_2] = \frac{1}{2} (\sigma_i^2 + \mu_i^2 - 1 - \log(\sigma_i^2)). \quad (4.5.33)$$

Now we can write down the full ELBO that the VAE will aim to maximize,

$$\text{ELBO} = -\text{MSE}(x, p(x|z)) - \sum_{i=0}^n \frac{1}{2} (\sigma_i^2 + \mu_i^2 - 1 - \log(\sigma_i^2)). \quad (4.5.34)$$

This can now be transformed into a loss function  $L_{\text{VAE}}$  to be minimized by setting  $L_{\text{VAE}} = -\text{ELBO}$ , leading to the complete VAE loss,

$$L_{\text{VAE}} = \text{MSE}(x, p(x|z)) + \sum_{i=0}^n \frac{1}{2} (\sigma_i^2 + \mu_i^2 - 1 - \log(\sigma_i^2)). \quad (4.5.35)$$

Intuitively this loss becomes minimal when the reconstructed output exactly matches the input and when the latent space is made up of Gaussians with  $\mu = 0$  and  $\sigma = 1$ . These two goals are, however, contradictory. If the latent space is perfectly regularized, it will be indistinguishable from Gaussian noise. Such a latent space would no longer contain any information, making the reconstruction task impossible. Therefore VAE training involves balancing the regularization loss with the reconstruction loss.

The end result is a setup that can map the input data onto a nearly Gaussian latent space and that can map this latent space back to the data space. One can now exploit this to generate new data points by drawing numbers from a Gaussian with  $\mu = 0$ ,  $\sigma = 1$ , and passing those through the decoder.

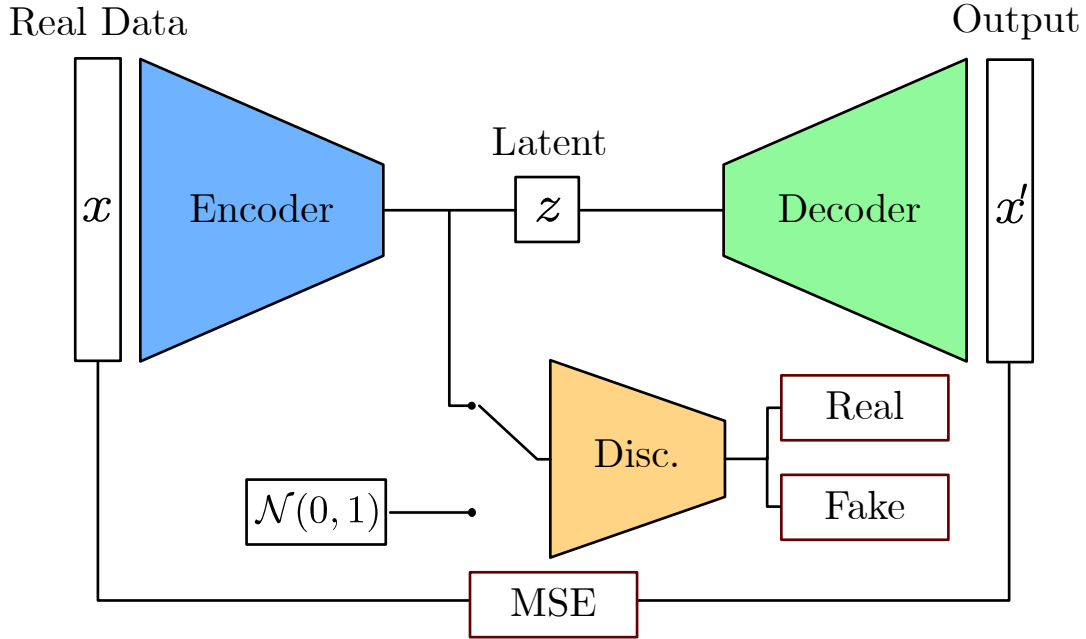


Figure 4.7: Schematic view of an AAE setup. The blue model shows the encoder  $E$ , the green model is the decoder  $D$  and the orange model is the latent discriminator  $A$ . For brevity, the encoder output  $D(z)$  is titled  $x'$ .

## 4.6 Adversarial Autoencoders

The VAE introduced in Section 4.5 demonstrated how the classical AE can be modified into a generative model through the addition of a latent space regularization. However, by construction, a VAE is limited to a tractable latent space, such as a multi-dimensional Gaussian. The Adversarial Autoencoder (AAE) [130] setup enables the use of more complex latent space distributions through the use of a GAN-like adversarial network.

The underlying structure is that of an AE, consisting of an encoder  $E$ , a decoder  $D$ , and an MSE-like reconstruction loss. Now let the target latent space be any distribution  $t(x)$  that can be sampled. In order to regularize the latent space produced by the encoder  $E(x)$  to be close to the target distribution, one trains an adversarial network  $A$  to distinguish between samples from the target distribution and samples produced by the encoder using the cross entropy loss function

$$-\mathbb{E}_{x \sim t_x} [\log(A(x, \theta_a))] - \mathbb{E}_{x \sim p_x} [\log(1 - A(D(x, \theta_d), \theta_a))] . \quad (4.6.1)$$

Much like how the discriminator in a GAN setup can be used to measure the difference between the distribution of the training data and the one produced by the generator, this adversarial network can be used to gauge how far the latent space is from the target distribution. Figure 4.7 shows the AAE setup, with the adversarial network used to regularize the latent space.

In order to minimize this distance, one can add the following term to the AE loss

$$\mathbb{E}_{x \sim p_x} [\log(1 - A(E(x, \theta_d), \theta_a))] . \quad (4.6.2)$$

This results in the AAE loss for the encoder and decoder,

$$L_{\text{AAE}} = \mathbb{E}_{x \sim p_x} [(x - D(E(x, \theta_e), \theta_d))^2] + \mathbb{E}_{x \sim p_x} [\log(1 - A(D(x, \theta_d), \theta_a))] . \quad (4.6.3)$$

Compared to the VAE, the AAE allows for an unrestrained choice of latent space, which can be advantageous when working with a data set that cannot be easily mapped to a Gaussian distribution for topological reasons. However, it adds significant difficulty to the training process as the fickle nature of the adversarial network makes balancing the reconstruction and regularization loss highly non-trivial.

## 4.7 The BIB-AE

The Bounded Information Bottleneck Autoencoder (BIB-AE) [131] was developed as an overarching theoretical model to investigate several generative approaches under the information bottleneck (IB) principle [132]. This section will first cover the broad theoretical ideas outlined in the original paper, followed by our implementation of the concept into a model setup. The derivation of the theoretical principles of the BIB-AE follows Reference [131], while the discussion of the implementation follows previously published work in Reference [2].

The IB approach argues that every deep learning approach can be seen as a trade-off between preserving and discarding information [133]. For example, a classification task can be understood as the classifier network trying to condense as much information as possible into the label prediction, while discarding irrelevant noise. An intuitive generative example is a VAE, where the bottleneck is explicitly built into the setup.

For such a VAE-like unsupervised IB model, let us assume we have a true distribution  $p_x(x)$ , from which a training set  $\{x_m\}_{m=1}^N$  is drawn. Here  $N$  refers to the total number of points in the training set and every  $x$  is an  $n$ -dimensional vector with  $x \in \mathbb{R}^n$ .  $m$  is a proper index associated with each  $x$ , meaning every  $x$  has exactly one  $m$  assigned to it.

As discussed in Section 4.5, the goal of the encoder part of the IB model is to map the data to a latent space  $z$ , using the approximate mapping  $q_\phi(z|x)$ , where in this case  $\phi$  are the parameters of the encoder network. Under the IB principle, this can be formulated as

$$\min_{\phi: I(\mathbf{Z}; \mathbf{X}) \geq I_x} I_\phi(\mathbf{X}; \mathbf{Z}) , \quad (4.7.1)$$

where  $\mathbf{X}$  is a vector of the training set and  $\mathbf{Z}$  is a vector of the latent space  $z$ . In this formulation,  $I_\phi(\mathbf{X}; \mathbf{Z})$  describes the mutual information between an input training sample and a latent vector for a given set of encoder parameters  $\phi$ .

The mutual information between random variables  $\mathbf{X}$  and  $\mathbf{Y}$  is defined as the relative entropy between the joint distribution  $p(x, y)$  and the product of the two marginal distributions  $p(x)p(y)$  [134]. This can be expressed using the KL-divergence as

$$I(\mathbf{X}; \mathbf{Y}) = \text{KL}[p(x, y), p(x)p(y)] \quad (4.7.2)$$

$$= \mathbb{E}_{p(x, y)} \left[ \log \left( \frac{p(X, Y)}{p(X)p(Y)} \right) \right]. \quad (4.7.3)$$

This mutual information is minimized, which corresponds to training the encoder to compress the inputs into a latent representation by removing some of the information. This minimization is, however, constrained by the expression  $I(\mathbf{Z}; \mathbf{X}) \geq I_x$ , where  $I(\mathbf{Z}; \mathbf{X})$  is once again the mutual information between an input and a latent vector, and  $I_x$  is the amount of information needed to achieve the given task of reconstructing  $X$ . Therefore the Equation (4.7.1) can be understood as the encoding network maximally compressing the input by minimizing the information in the latent space while ensuring that the information that does remain in the latent space is sufficient for the decoder to reconstruct the original input.

This can be formulated as a Lagrangian optimization problem, where one aims to minimize the term

$$\mathcal{L}(\phi) = I_\phi(\mathbf{X}; \mathbf{Z}) - \beta I(\mathbf{Z}; \mathbf{X}), \quad (4.7.4)$$

where  $\beta$  is a relative scaling between the compression and reconstruction criteria.

Using the definition of the mutual information in Equation 4.7.3, as well as the fact that the mutual information is symmetric, one can rewrite the first expression in the Lagrangian  $I_\phi(\mathbf{X}; \mathbf{Z})$  as

$$I_\phi(\mathbf{X}; \mathbf{Z}) = \mathbb{E}_{q_\phi(z, x)} \left[ \log \left( \frac{q_\phi(z, x)}{q_\phi(z)p_x(x)} \right) \right], \quad (4.7.5)$$

with the marginal distribution of  $z$  defined as  $q_\phi(z) = \mathbb{E}_{p_x(x)} q_\phi(z, x)$ . Using Equation (4.5.1) we can express the joint probability as  $q_\phi(z, x) = q_\phi(z|x)p_x(x)$ . This can be used to modify the mutual information expression as

$$I_\phi(\mathbf{X}; \mathbf{Z}) = \mathbb{E}_{q_\phi(z, x)} \left[ \log \left( \frac{q_\phi(z|x)p_x(x)}{q_\phi(z)p_x(x)} \right) \right] \quad (4.7.6)$$

$$= \mathbb{E}_{q_\phi(z, x)} \left[ \log \left( \frac{q_\phi(z|x)}{q_\phi(z)} \right) \right]. \quad (4.7.7)$$

Now one can use variational inference to approximate  $q_\phi(z)$  using a new distribution  $p_\theta(z)$ , this allows the above expression to be rewritten as

$$I_\phi(\mathbf{X}; \mathbf{Z}) = \mathbb{E}_{q_\phi(z, x)} \left[ \log \left( \frac{q_\phi(z|x)p_\theta(z)}{q_\phi(z)p_\theta(z)} \right) \right] \quad (4.7.8)$$

$$= \mathbb{E}_{q_\phi(z, x)} \left[ \log \left( \frac{q_\phi(z|x)}{p_\theta(z)} \right) \right] - \mathbb{E}_{q_\phi(z, x)} \left[ \log \left( \frac{q_\phi(z)}{p_\theta(z)} \right) \right]. \quad (4.7.9)$$



Making use of  $q_\phi(z, x) = p_x(x)q_\phi(z|x)$  to rewrite the expectation value in the first expression as

$$\mathbb{E}_{q_\phi(z,x)} \left[ \log \left( \frac{q_\phi(z|x)}{p_\theta(z)} \right) \right] = \mathbb{E}_{p_x(x)} \left[ \mathbb{E}_{q_\phi(z|x)} \left[ \log \left( \frac{q_\phi(z|x)}{p_\theta(z)} \right) \right] \right]. \quad (4.7.10)$$

The second expectation value in this expression is equal to the KL divergence between  $q_\phi(z|x)$  and  $p_\theta(z)$  for  $X = x$ ,

$$\mathbb{E}_{q_\phi(z|x)} \left[ \log \left( \frac{q_\phi(z|x)}{p_\theta(z)} \right) \right] = \mathbb{E}_{q_\phi(z|x)} \left[ \text{KL}(q_\phi(z|X=x), p_\theta(z)) \right]. \quad (4.7.11)$$

Similarly one can use the fact that the second term does not depend on  $x$  to express the expectation value as another KL divergence,

$$\mathbb{E}_{q_\phi(z,x)} \left[ \log \left( \frac{q_\phi(z)}{p_\theta(z)} \right) \right] = \mathbb{E}_{q_\phi(z)} \left[ \log \left( \frac{q_\phi(z)}{p_\theta(z)} \right) \right] = \text{KL}(q_\phi(z), p_\theta(z)). \quad (4.7.12)$$

The second mutual information term  $I(\mathbf{Z}; \mathbf{X})$  can be bounded from below [131] by

$$I(\mathbf{Z}; \mathbf{X}) \geq I_{\theta,\phi}(\mathbf{Z}; \mathbf{X}), \quad (4.7.13)$$

where  $\theta$  are the parameters of the decoder network  $p_\theta(x|z)$ , that approximates the true mapping from latent to data space  $p(x|z)$  and  $I_{\theta,\phi}(\mathbf{Z}; \mathbf{X})$  is defined as

$$I_{\theta,\phi}(\mathbf{Z}; \mathbf{X}) = -\mathbb{E}_{p_x(x)} \left[ \log(p_x(x)) \right] + \mathbb{E}_{p_x(x)} \left[ \mathbb{E}_{q_\phi(z|x)} \left[ \log(p_\theta(x|z)) \right] \right]. \quad (4.7.14)$$

Similar to the first mutual information term this can be reformulated as

$$I_{\theta,\phi}(\mathbf{Z}; \mathbf{X}) = \mathbb{E}_{q_\phi(z|x)} \left[ \mathbb{E}_{p_x(x)} \left[ \log \left( \frac{p_\theta(x|z)}{p_x(x)} \right) \right] \right] \quad (4.7.15)$$

$$= \mathbb{E}_{q_\phi(z|x)} \left[ \mathbb{E}_{p_x(x)} \left[ \log \left( \frac{p_\theta(x|z)p_\theta(x)}{p_x(x)p_\theta(x)} \right) \right] \right] \quad (4.7.16)$$

$$= -\mathbb{E}_{p_x(x)} \left[ \log(p_\theta(x)) \right] - \mathbb{E}_{p_x(x)} \left[ \log \left( \frac{p_x(x)}{p_\theta(x)} \right) \right] \quad (4.7.17)$$

$$+ \mathbb{E}_{q_\phi(z|x)} \left[ \mathbb{E}_{p_x(x)} \left[ \log(p_\theta(x|z)) \right] \right].$$

Now one can apply a second lower bound, using the fact that  $-\mathbb{E}_{p_x(x)} \left[ \log(p_\theta(x)) \right]$  is always greater than or equal to zero. This second lower bound  $I_{\theta,\phi}(\mathbf{Z}; \mathbf{X})^L \leq I_{\theta,\phi}(\mathbf{Z}; \mathbf{X})$  is then given by,

$$I_{\theta,\phi}(\mathbf{Z}; \mathbf{X})^L = -\mathbb{E}_{p_x(x)} \left[ \log \left( \frac{p_x(x)}{p_\theta(x)} \right) \right] + \mathbb{E}_{q_\phi(z|x)} \left[ \mathbb{E}_{p_x(x)} \left[ \log(p_\theta(x|z)) \right] \right] \quad (4.7.18)$$

Similar to Equation (4.7.11) the first term of  $I_{\theta,\phi}(\mathbf{Z}; \mathbf{X})^L$  can be written as a KL-divergence between  $p_x(x)$  and  $p_\theta(x)$ , resulting in the final form:

$$I_{\theta,\phi}(\mathbf{Z}; \mathbf{X})^L = -\text{KL}(p_x(x), p_\theta(x)) + \mathbb{E}_{q_\phi(z|x)} \left[ \mathbb{E}_{p_x(x)} \left[ \log(p_\theta(x|z)) \right] \right] \quad (4.7.19)$$

Using this one can now define the BIB-AE as an AE setup based on the IB principle. Its Lagrangian can be summarized as

$$\begin{aligned} \mathcal{L}_{\text{BIB-AE}} = & \underbrace{\mathbb{E}_{q_\phi(z|x)} \left[ \text{KL}(q_\phi(z|X=x), p_\theta(z)) \right]}_{\mathbf{A}} + \underbrace{\text{KL}(q_\phi(z), p_\theta(z))}_{\mathbf{B}} \\ & + \underbrace{\mathbb{E}_{q_\phi(z|x)} \left[ \mathbb{E}_{p_x(x)} \left[ \log(p_\theta(x|z)) \right] \right]}_{\mathbf{C}} - \underbrace{\text{KL}(p_x(x), p_\theta(x))}_{\mathbf{D}} . \end{aligned} \quad (4.7.20)$$

The four terms **A**, **B**, **C**, and **D** each fulfill their own purpose within the BIB-AE framework. **A** and **B** force the encoder to minimize the information contained in the latent space by pushing the encoded latent space to match the targeted latent space  $p_\theta(z)$ . Here **A** can be understood as a direct KL-divergence between the encoded and target latent space, while **B** can be seen as a sample-based comparison method such as an adversarial network or a Maximum Mean Discrepancy (MMD, also see Section 6.3 for details) [135] term, that approximates the KL-divergence between the produced latent space  $q_\phi(z)$  and the target  $p_\theta(z)$ . Terms **C** and **D** ensure the information in the latent space is still sufficient to reconstruct data that matches the training data distribution  $p_x(x)$ . This is achieved either through a direct calculation of the log-likelihood, such as in **C**, or through another sample-based method to estimate the KL-divergence in **D**.

As such, the previously explored generative models can be expressed in these terms. A *VAE* can be understood to use a direct calculation of the KL-divergence in the latent space, corresponding to **A**, and its MSE reconstruction loss corresponds to the log-likelihood in **C**. An *AAE* similarly uses **C** to facilitate its reconstruction, and its adversarial latent regularization is captured in **B**. A standard *GAN* makes use of the adversarial comparison between decoder/generator output, which is described by **D**.

These observations lead to our practical BIB-AE implementation, as they allow us to connect the information-theoretical terms to components of generative networks:

- **A**: latent space KL-divergence term
- **B**: latent space discriminator or MMD term
- **C**: data space MSE term
- **D**: data space discriminator

Figure 4.8 shows the full model with all its components. An intuitive way to view our BIB-AE implementation is as an expanded VAE architecture. At its core, it consists of an encoder-decoder structure, with a latent space consisting of a set of Gaussian distributions, defined by a vector of means and a vector of widths. The generation process of the BIB-AE is closely related to a

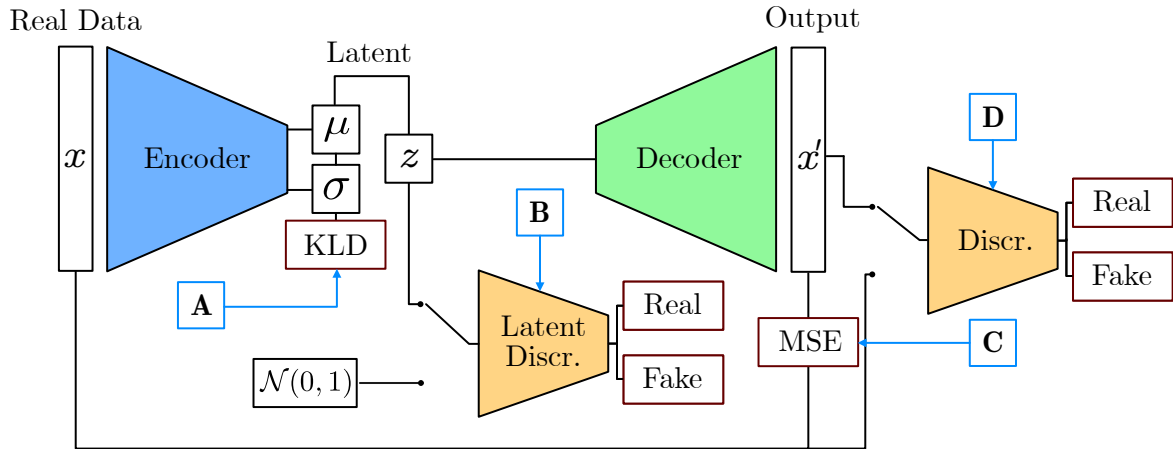


Figure 4.8: The full BIB-AE setup, according to our implementation. The encoder and decoder are depicted in blue and green respectively, and the two adversarial networks are depicted in orange. The letters **A** through **D** indicate which model part corresponds to which mathematical term.

VAE, in that a noise vector is sampled from a  $\mu = 0, \sigma = 1$  Gaussian, and then passed through the decoder, in order to generate new data.

During training, the input data is mapped to this latent space by the encoder. The latent space is regularized to be close to a  $\mu = 0, \sigma = 1$  Gaussian through a KL-divergence, similar to a VAE, and an adversarial network, similar to an AAE. These two terms act complementary to one another. The KL-divergence is very well suited to reduce the deviations between latent space dimensions and the target latent space, however, it only captures the average overall latent dimensions. This can result in a situation where the majority of latent dimensions are well regularized, but a small number of dimensions still show large deviations. The information encoded in individual latent dimensions is proportional to their deviation from the target latent distribution. Therefore these few, large-deviation dimensions will contain the bulk of the information in the latent space. As such they will be the most important decoder inputs for the reconstruction task. This presents a problem during generation, as the most important inputs to the decoder will have the most significant deviations compared to what the decoder was trained to expect. A preferable case would be to have the information evenly distributed across all latent space dimensions. The adversarial network can be trained to examine individual latent space distributions. Furthermore, it will inherently put higher penalties on larger deviations, as these make the classification task into “real” and “fake” latent space samples very simple. Therefore, the adversarial network can work in conjunction with the KL-divergence to produce a latent space with a more even information distribution.

The encoded latent samples are then passed into the decoder, which is tasked with reconstructing the encoder network input. This reconstruction is once again facilitated by two components, an MSE comparing input and output directly, and an adversarial discriminator trained to distinguish between “real” data samples and “fake” reconstructed samples. These two terms once again complement one another. Generally speaking, the discriminator loss of a GAN will result

in higher quality generated data, than the simple MSE of a VAE [136]. This stems from the fact that a discriminator can learn to look for complex features and correlations in the generated data, while the MSE can only compare individual pixel deviations. However, a discriminator takes individual decoder outputs and determines whether they look like a realistic data point. It does not take into account whether the output looks similar to its specific, corresponding encoder input. Therefore, if one were to only use a discriminator to evaluate the decoder output of the BIB-AE, there would be nothing to enforce a similarity between the input and the reconstruction result. This would remove the requirement for the latent space to contain any relevant information, contradicting the underlying IB idea. In principle, such a discriminator-only setup would be equivalent to a GAN, mapping informationless, Gaussian noise to the data space. The MSE loss however inherently requires the latent space to contain information, as the only way to completely minimize the MSE loss is for the input and output to match up exactly. Therefore, the combination of MSE loss and discriminator allows for both the preservation of information in the latent space, as well as for the generation of high-quality images.

The BIB-AE setup represents the core model used for fast simulation in this work, and its application for the simulation of photon and pion calorimeter showers will be explored in subsequent Chapters 6, 7, and 8.

## 4.8 Normalizing Flows

AE-based generative models solve the problem of defining a generative loss function by having an encoder map the training data to a well-understood latent distribution, from which a decoder then maps back to the data space. The decoder is effectively trained to be an approximate inverse of the encoder and vice versa. However, because encoder and decoder are still two distinct networks, this relation will always have at least slight imperfections.

An arguably preferable approach would be to have a single, invertible, neural network, that can act as a mapping from both the latent to data space and back from the data to latent space. This is the underlying idea of Invertible Neural Networks (INN) or Normalizing Flows.

The discussion of Normalizing Flows in this section follows References [137,138]. The underlying principle of Normalizing Flows is that, if one has a sufficiently well-behaved data distribution  $p_x$  with samples  $\mathbf{x} \in \mathbb{R}^d$  then one can model  $p_x$  by expressing  $\mathbf{x}$  as a transformation  $T$  of  $\mathbf{z}$ , where  $\mathbf{z}$  is a sample from a well understood latent distribution  $p_z(\mathbf{z})$ .

$$\mathbf{x} = T(\mathbf{z}) \tag{4.8.1}$$

If  $T$  is a diffeomorphism, i.e. an invertible transformation that is differentiable almost everywhere in both its forward and inverse direction, then  $p_x$  can be expressed using the change of variables formula [139,140],

$$p_x(\mathbf{x}) = p_z |\det J_T(z)|^{-1} \text{ or,} \tag{4.8.2}$$

$$p_x(\mathbf{x}) = p_{T^{-1}(\mathbf{x})} |\det J_{T^{-1}}(\mathbf{z})|, \tag{4.8.3}$$

where  $J_T(\mathbf{x}) = \frac{\partial T}{\partial \mathbf{z}}$  is the Jacobian of  $T$ . This means if one has a sufficiently expressive diffeomorphism  $T$ , then this transformation can be used to warp the latent distribution in a way that

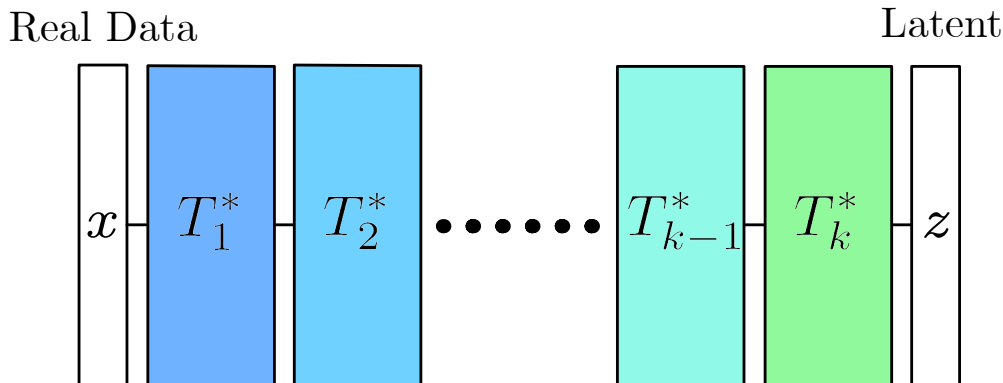


Figure 4.9: A flow setup mapping from data to latent space using consecutive invertible transformations  $T_i^*$ , with  $i$  running from 1 to the total number of transformations  $k$ .

models the desired data distribution. The determinant of the Jacobian measures the volume change around  $z$  caused by  $T$ . In practice, defining highly expressive single transformations can be difficult. However, the product of two diffeomorphisms is once again a diffeomorphism, as both functions can be separately inverted and differentiated. Therefore, one can construct an arbitrarily complex transformation  $T$  by chaining multiple simple transformations  $T_{1,\dots,k}$  together

$$T = T_1 \circ T_2 \circ \dots \circ T_k. \quad (4.8.4)$$

For such a composite transformation the determinant of the Jacobian can be calculated separately as

$$\det(J_{T_2 \circ T_1})(\mathbf{x}) = \det J_{T_2}(T_1(\mathbf{x})) \cdot \det(J_{T_1}(\mathbf{x})). \quad (4.8.5)$$

In a normalizing flow, this property is immensely valuable, as it allows the use of several individual network layers, each encoding a simple transformation  $T_{1,\dots,k}^*$ , which can then be combined to form the full normalizing flow network with parameters  $\theta$  that models the transformation  $T^*(\theta)$ . Such a composite flow model made up of individual transformations is shown in Figure 4.9.

This model can now be trained by modifying  $\theta$  such that  $T^*(\theta)$  approximates the desired mapping  $T$  between data and latent space. If  $q_x(x, \theta)$  is the data distribution given by the flow, and  $p_x(x)$  is the training distribution, then the model can be optimized by minimizing the KL-divergence between the two distributions, leading to the flow loss

$$\mathcal{L}_{\text{Flow}} = \text{KL}(p_x(\mathbf{x}), q_x(\mathbf{x}, \theta)) \quad (4.8.6)$$

$$= -\mathbb{E}_{p_x(\mathbf{x})} \left[ \log \left( \frac{q_x(\mathbf{x}, \theta)}{p_x(\mathbf{x})} \right) \right] \quad (4.8.7)$$

$$= -\mathbb{E}_{p_x(\mathbf{x})} [\log(q_x(\mathbf{x}, \theta)) - p_x(\mathbf{x})] \quad (4.8.8)$$

$$= -\mathbb{E}_{p_x(\mathbf{x})} [\log(q_x(\mathbf{x}, \theta))] + \text{const.} \quad (4.8.9)$$

For the generative models discussed in previous sections, this probability density is not directly tractable. Therefore, one would not be able to compute this final expression, requiring the use of workarounds to estimate the divergence, such as the adversarial network of the GAN.

For a Flow model however one can exploit the invertibility of the model to express  $\log q_x(x, \theta)$  through  $T$  and the latent distribution

$$\mathcal{L}_{\text{Flow}} = -\mathbb{E}_{p_x(\mathbf{x})}[\log(p_z(T^{*-1}(\mathbf{x}, \theta)))] + \log(|\det J_{T^{-1}}(\mathbf{x}, \theta)|) + \text{const.} . \quad (4.8.10)$$

For a given set of training data with  $N$  points  $\mathbf{x}_n, n \in 1, \dots, N$  the expectation value over  $p_x(\mathbf{x})$  can be approximated by the expectation value over the sample, assuming the sample is large enough to be statistically representative. This results in the training loss for the flow

$$\mathcal{L}_{\text{Flow}} = \frac{1}{N} \sum_{n=1}^N \log(p_z(T^{*-1}(\mathbf{x}_n, \theta))) + \log(|\det J_{T^{-1}}(\mathbf{x}_n, \theta)|) + \text{const.} . \quad (4.8.11)$$

Calculating this expression only requires the evaluation of the latent prior distribution  $p_z(z)$ . As the exact latent distribution is arbitrary, one can choose an easily evaluated distribution such as a univariate Gaussian. Therefore, the normalizing flow approach can directly calculate the negative log-likelihood, without any workaround. This leads to flows having an especially stable training behavior compared to other generative approaches such as GANs.

In order to use the trained flow as a generator, one needs to sample from the latent distribution and pass these samples through the inverse transformation.

The desirable properties of a normalizing flow are contingent on the use of invertible transformations. Building a neural network to be invertible is a highly non-trivial task, and a multitude of approaches exist. The two main ones used in this work will be covered here.

## Coupling layers

Generally speaking, neural network layers cannot be directly inverted as their corresponding weight matrices are not necessarily invertible. Coupling layers [141] are specifically designed to circumvent this problem.

In a coupling layer the  $d$ -dimensional input vector  $\mathbf{x} \in \mathbb{R}^d$  is split into two vectors  $\mathbf{x}_A \in \mathbb{R}^{d_A}$  and  $\mathbf{x}_B \in \mathbb{R}^{d_B}$ , with  $d = d_A + d_B$ . One then defines an invertible function  $f(\theta)$  with the parameters  $\theta$ . A common example is the linear affine transformation. It uses two parameter vectors,  $\theta = \mathbf{s}, \mathbf{t}$ , which are used to scale and shift the inputs,

$$h(\mathbf{x}, (\mathbf{s}, \mathbf{t})) = \mathbf{x} \otimes \mathbf{s} + \mathbf{t} , \quad (4.8.12)$$

where  $\otimes$  is the element-wise multiplication between vectors. The inverse of this function can be directly calculated with

$$h(\mathbf{x}, (\mathbf{s}, \mathbf{t}))^{-1} = \frac{\mathbf{x} - \mathbf{t}}{\mathbf{s}} . \quad (4.8.13)$$

Figure 4.10 indicates how one then uses the second part of the split vectors to define the parameters to apply the transformation to the first split, resulting in the first layer output  $\mathbf{u}_A$ .

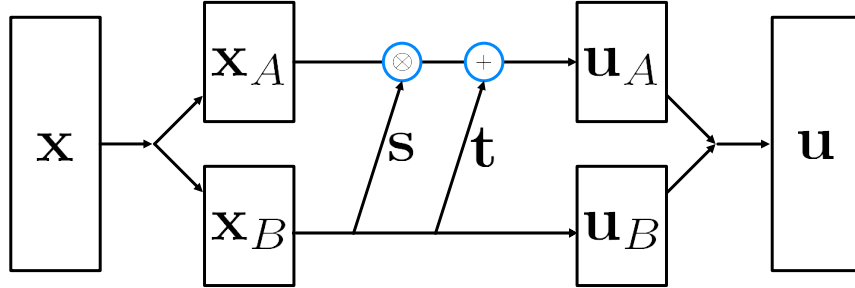


Figure 4.10: Schematic depiction of a coupling layer. The transformation applied to  $\mathbf{x}_A$  only depends on  $\mathbf{x}_B$ , making the layer invertible.

The second output  $\mathbf{u}_B$  is the unchanged input  $\mathbf{x}_B$ . This gives the full input to output mapping of a coupling layer

$$\mathbf{u}_A = h(\mathbf{x}_A, (\mathbf{s}(\mathbf{x}_B), \mathbf{t}(\mathbf{x}_B))) = \mathbf{x}_A \otimes \mathbf{s}(\mathbf{x}_B) + \mathbf{t}(\mathbf{x}_B) \quad (4.8.14)$$

$$\mathbf{u}_B = \mathbf{x}_B . \quad (4.8.15)$$

Since  $\mathbf{x}_B$  is equal to  $\mathbf{u}_B$  it is available in both the forward and inverse direction. Therefore, we can define the inverse of the coupling layer as

$$\mathbf{x}_A = h^{-1}(\mathbf{u}_A, (\mathbf{s}(\mathbf{u}_B), \mathbf{t}(\mathbf{u}_B))) = \frac{\mathbf{u}_A - \mathbf{t}(\mathbf{u}_B)}{\mathbf{s}(\mathbf{u}_B)} \quad (4.8.16)$$

$$\mathbf{x}_B = \mathbf{u}_B . \quad (4.8.17)$$

The scaling  $\mathbf{s}(\mathbf{x})$  and shifts  $\mathbf{t}(\mathbf{x})$  are generally implemented as fully connected neural networks with  $d_B$  input nodes and  $d_A$  output nodes. The exact splitting of  $\mathbf{x}$  into  $\mathbf{x}_A$  and  $\mathbf{x}_B$  is arbitrary, however, the most common approach is to split  $\mathbf{x}$  into two halves, as an uneven split would place greater importance on certain input dimensions.

The coupling layer defined in Equation (4.8.15) is sufficient to build a flow network. However, most implementations add a second transformation to the previously untouched part of the input. This ensures every variable is modified by the transformation. This is shown in Figure 4.11. Such an approach can effectively be understood as the combination of two coupling transformations, meaning this type of coupling block is also invertible.

**Autoregressive Layers:** Similar to coupling layers, autoregressive layers [142] use transformations that are independent of their direct input to allow for invertibility. If one defines  $h_i(\theta)$  as the transformation with parameters  $\theta$ , applied to the  $i$ -th dimension of the input  $x_i$ , then in the autoregressive scheme the first transformation is constant

$$u_1 = h_1(x_1) , \quad (4.8.18)$$

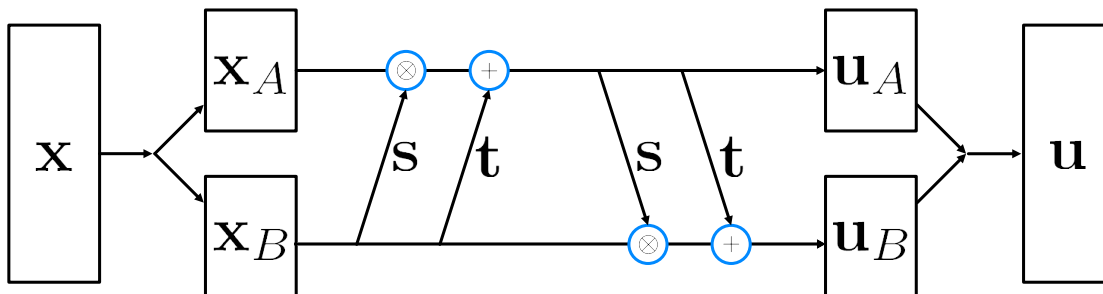


Figure 4.11: A double version of the coupling layer, derived by taking Figure 4.10 and repeating the transformation principle on  $\mathbf{x}_B$ .

and every subsequent transformation is dependent on the previous input dimensions

$$u_i = h_i(x_i, (x_{i-1}, \dots, x_1)), \text{ for } i \in \{2, \dots, d\}. \quad (4.8.19)$$

This scheme means that the inverse of the first dimension is always known, as its transformation is constant. The inverse of the second dimension can be constructed using the known first dimension, which can then in turn be used to invert the third dimension, and so forth.

The exact form of  $h_i$  is arbitrary, as long as it remains a diffeomorphism - for example the scale and shift introduced for the coupling layer can be used. The individual transformations are once again implemented using a neural network.

The forward pass of an autoregressive flow can be implemented using a single network execution, the inputs of which are masked to ensure the dependence scheme outlined above. This specific implementation is known as a masked autoregressive layer (MAF) [143]. The masked approach allows the forward direction of the flow to be calculated quickly, however the inverse direction requires the sequential computation of the individual dimensions, resulting in a significant slowdown.

Overall, MAF layers have more flexibility compared to coupling layers, as they do not have the strict separation into two input groups. However, in practical generative applications, the slower inverse computation results in a significantly slower sampling procedure when using MAF layers.

## Permutation Layers

Several coupling or MAF layers can be combined to form an expressive normalizing flow. However, a direct stacking of the transformations can have unintended consequences, as certain input dimensions will be treated differently. For example, a stack of coupling layers would never allow the dimensions within the same halves to interact with each other and no correlations between the two halves could be learned.

For this reason, every flow network uses permutation layers to modify the order of the dimensions after every transformation. As these permutations are fixed, they can be easily inverted.



The most common permutations are reverse permutations, which flip the order of the dimensions, or random permutations, that randomly choose a permutation scheme at initialization.

## 4.9 Conditional Generative Models

So far, all generative approaches that have been discussed aim to generate new data samples from random noise. For this purpose, the exact nature of the generated points is not relevant, beyond the requirement that they are in line with the training data distribution. However, in a practical application of generative models, it is often helpful or even required, to produce data points with specific properties.

An instructive example can be made with the MNIST handwritten number data set introduced in Section 4.1. Each data point consists of one image  $x$ , and one associated class label  $c \in \{0, \dots, 9\}$ . If one trains a standard generative model on this data set, one will end up with a generator  $G$  that maps a latent space sample  $z \in Z$  to a new sample  $x_{\text{gen}} = G(z)$ . If the model is well trained this sample should have the appearance of a handwritten digit and have the same likelihood of depicting the numbers 0 to 9 as the data set. However, from the view of most practical applications, generating a random handwritten digit is of little use. It would be preferable to generate a specific digit, as this would for example allow for a given number to be converted into a handwritten version. Conditional generative models exist to address precisely this problem. A conditional generator  $G_{\text{cond}}(z, c)$  accepts an additional input  $c$  that specifies the class of the object that is to be generated. While the exact appearance of a given sample will still have some randomness due to the input noise, all samples  $x_c = G_{\text{cond}}(z, c)$  produced with a given label  $c$  will still be part of that class. In the MNIST example this would mean that a generator tasked with producing “nines” would produce several different images, however, all of them would still depict the number “nine”.

All generative models covered in this chapter can be transformed into conditional generative models. The specific methods of how this can be achieved will be indicated below.

**GANs/WGANs:** The class label is given as an additional input to both the generator and discriminator/critic network. During training, the generator is passed a label that is either taken from the training data or randomly sampled <sup>2</sup>. The discriminator/critic is then trained to classify the combination of training data and their corresponding labels as “real” and the combination of generated data and the labels used during generation as “fake”. This allows the discriminator/critic to learn the correlation between features in a given data point and its corresponding label. Therefore, the discriminator/critic will be able to detect instances where the generated data does not match up with its given label, forcing the generator to learn to produce samples that fit the labels.

**VAEs/AEs:** The class label is added as an input to the encoder and decoder networks. This means the decoder always has access to the information contained in the label, regardless of the latent encoding of the encoder. Therefore, the encoder, which aims to minimize the amount of unimportant information in the latent space is incentivized to not include any label information in the latent space, as this would be redundant. This, in turn, forces the decoder to learn to directly make use of the class label, as the label contains information required to correctly reconstruct the input. It should be noted that this is a less strict conditioning compared to GAN

---

<sup>2</sup>In the case of the WGAN labels should always be taken from the current training batch, to ensure the interpolation between real and generated data performed during gradient penalty is sensible.

approaches, as there is no explicit penalty term for producing data that does not match the given label.

**BIB-AE:** The conditional BIB-AE setup is like the BIB-AE itself, a combination of the VAE and GAN conditioning approaches. The label is passed to the encoder, decoder, and adversarial network. Similarly to the VAE, the decoder is incentivized to make use of the class label for the reconstruction process. Additionally, the adversarial network will detect and penalize mismatches between labels and decoder outputs.

**Normalizing Flow:** During training, the class label is passed as an additional input to the networks that determine the transformation parameters. If the information contained in the label is relevant for a given transformation, the associated network will learn to make use of the label when mapping the data to the latent space. During generation, the label is once again passed to each network, which guides the mapping from latent to data space such that the resulting sample corresponds to an instance of the given label.

## Part II

# Generative Models in Particle Physics



## Chapter 5

# GANplification

The work presented in this chapter has been previously published as Reference [1] in collaboration with Anja Butter, Gregor Kasieczka, Benjamin Nachman, and Tilman Plehn. Several figures presented as well as the text are similar or identical to the content of this article. My contribution to the publication comprised implementation and optimization of the generative model, development of the quantile-MSE metric, writing sections of the paper, handling the peer-review processes, and addressing referee comments.

In HEP, generative models present an exciting method for fast simulation. The most common approach to this uses a classical simulation method to generate a training set. This is then used to train a generative model, from which the bulk of the samples is drawn. This approach leverages the fact that generative models can be evaluated significantly faster than classical simulations, in order to provide an overall speed up.

There is, however, one question that jeopardizes this approach and needs to be addressed first: *How many samples can be drawn from a generative model before one becomes limited by the statistics of the original training set?* The naive assumption would be *as many samples as the model was trained on*. If that were the case, it would severely limit the use of generative models for simulation, as any analysis that uses the generative samples would be no more accurate than if it had used the training data instead. However, the answer to the question is not necessarily that intuitive. Since a generative model does not simply learn individual data points but instead learns an underlying distribution, this can allow the model to interpolate between the points of the training data, potentially allowing it to reasonably produce more samples than it was initially trained on. If this is true, then such models remain viable for fast simulation.

This makes answering this question a high priority for anyone working on fast simulation with generative models. The concept of taking a limited data set and increasing the available number of points is often called *data amplification* [144]. However, little work has been done on generative models in specific. Therefore, we explore the statistical behavior of generative models using a set of small-scale toy models.

This chapter is organized as follows. Section 5.1 covers initial explorations using a 1-dimensional data set. This is extended to two and five dimensions in Sections 5.2 and 5.3. Section 5.4 presents the conclusions obtained from this exploration.

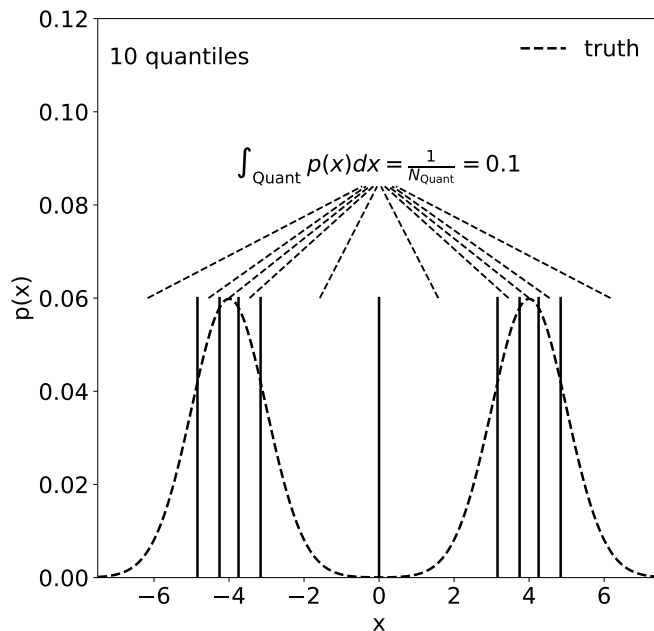


Figure 5.1: The camel back distribution used for the initial generative amplification investigation. Vertical black lines indicate the edges of the quantile regions.

## 5.1 1-Dimensional Data

We initially investigate a 1-dimensional distribution. The low dimensionality allows for fast training times and requires less complex models, allowing us to gain early insight. The training distribution is a *camel back*, consisting of two normalized Gaussians located symmetrically around zero, given by

$$p(x) = \frac{1}{2}\mathcal{N}_{4,1}(x) + \frac{1}{2}\mathcal{N}_{-4,1}(x). \quad (5.1.1)$$

This distribution was chosen since its disjoint nature can be challenging to learn for a generative network, despite being only one-dimensional. This makes the camel back a suitable proof of concept example. The dashed line in Figure 5.1 visualizes the camel back.

As our training data, we draw 100 points from the camel back distribution. While this number of training points is small compared to the tens- or hundreds of thousands usually used in HEP ML, it is still sufficient for a 1-dimensional problem.

The question of how many samples one can draw from a generative model is fundamentally linked to how well the underlying distribution is described by either the generative model or by the data used in training. If the model is more accurate at describing this underlying distribution than the training data alone, then it is reasonable to generate more points than were used for training.

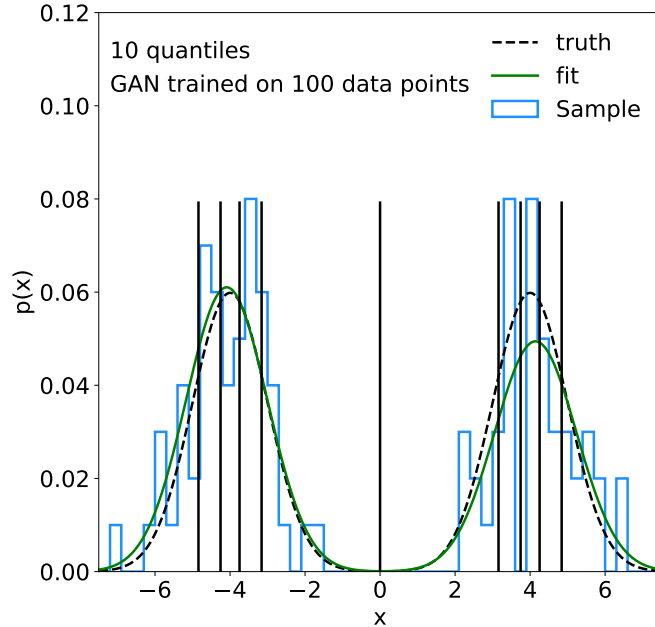


Figure 5.2: Camel back distribution. The blue histogram shows the 100 points in the training sample. The green line is the result of the parameter fit. The black vertical lines again indicate the quantile regions.

In order to estimate the agreement between a given sample of data, such as the training set, and the true underlying distribution given in Equation (5.1.1) we first divide the true distributions into  $N_{\text{quant}}$  quantiles. These quantiles are defined to each contain  $\frac{1}{N_{\text{quant}}}$  of the total probability of the distribution. An example of this with  $N = 10$  is shown by the vertical black lines in Figure 5.1. The quantile regions defined by the true distribution are then used to divide the training sample and the fraction of points in the  $j$ -th quantile region,

$$x_j = \frac{n_j}{n_{\text{total}}}, \quad (5.1.2)$$

is calculated, where  $n_j$  are the number of points in the  $j$ -th quantile and  $n_{\text{total}}$  the total number of points. Per definition the expected fraction in each quantile is  $\frac{1}{N_{\text{quant}}}$ . Therefore we can compare how well the sample agrees with the true distribution by calculating the mean squared difference between  $x_j$  and  $\frac{1}{N_{\text{quant}}}$ . This leads to what we define as the Quantile Mean Squared Error (MSE), given by

$$\text{MSE} = \frac{1}{N_{\text{quant}}} \sum_{j=1}^{N_{\text{quant}}} \left( x_j - \frac{1}{N_{\text{quant}}} \right)^2. \quad (5.1.3)$$

We initially aim to find an upper benchmark for how well a generative model could ideally perform. In essence, training a generative model on a data set is comparable to performing a

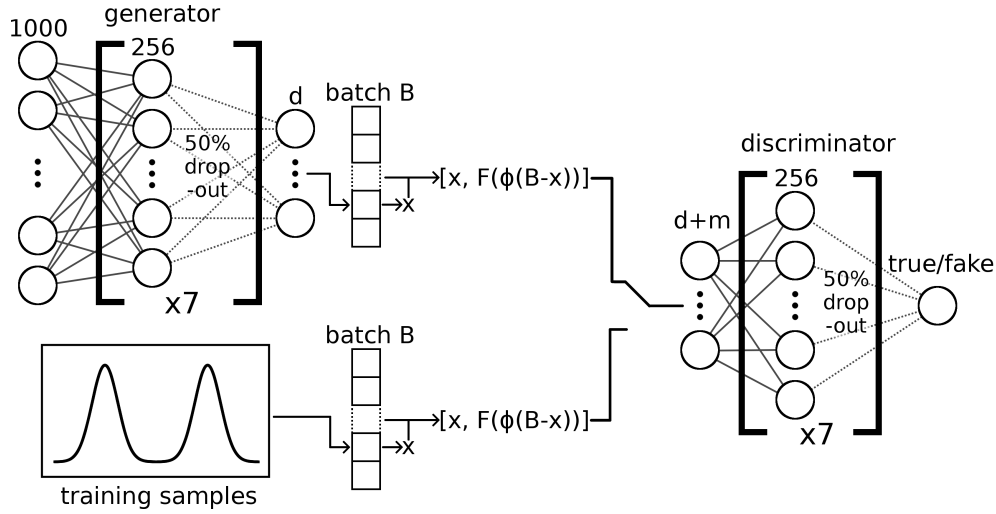


Figure 5.3: Schematic depiction of the GAN architecture that was used.  $\Phi$  and  $F$  are the embedding function and aggregation function used in the minibatch discrimination, respectively. Figure taken from [1].

parametric fit on that data. However, since the generative model is significantly more complex and universal, it is reasonable to assume that a parameter fit that uses the correct functional form will perform better than the generative model. Therefore our upper performance benchmark consists of a fit using the true camel back function with 5 parameters. These parameters are the means of the Gaussians  $\mu_1$ ,  $\mu_2$ , their widths  $\sigma_1$ ,  $\sigma_2$  and a parameter  $a_0$  determining the relative contribution of the two peaks. The specific function is given by

$$p(x) = a_0 \mathcal{N}_{\mu_1, \sigma_1}(x) + (1 - a_0) \mathcal{N}_{\mu_2, \sigma_2}(x). \quad (5.1.4)$$

This fit is performed using the PYTHON packages IMINUIT [145] and PROBFIT [146]. Figure 5.2 shows the result of this fit in green. Further, the figure also shows a histogram of the training data in blue.

The generative model used is based on a standard GAN architecture, as described in Section 4.2. Several adaptations are made to the model to accommodate the small size of the training set, however, we aim to keep the model as general as possible, and therefore avoid any network optimizations specific to the camel back distribution.

The generator network of the GAN is a 7 layer fully connected network (FCN) with 1 output node, 256 hidden nodes in each layer, and 1000 input nodes. The dense layers are interspersed with dropout [111] layers with a 50% dropout factor. These layers reduce the tendency of the model to overfit to the small number of training data. These dropout layers are kept active during data generation. Further, to aid the model in learning a smooth distribution we employed ELU [115] activation functions. The generator input consists of 1000 numbers drawn from random noise. GANs commonly use Gaussian noise as generator input, however, to avoid potentially biasing our model we chose the input noise to be drawn from a uniform distribution in the range  $[-1, 1]$ .



In order to avoid mode collapse, the discriminator network makes use of minibatch discrimination [147]. This means the discriminator not only takes the individual  $d$ -dimensional data points  $x \in \mathbb{R}^d$  into account, but also the entire  $n$ -length batch  $B \in \mathbb{R}^{d,n}$  that contains  $x$ . This allows the discriminator to have a direct handle on whether the spread of generated points matches that of the training data. The discriminator first calculates the pairwise distance between  $x$  and the entire batch. This distance  $x - B$  is then passed through a deepset-inspired [148, 149] embedding function  $\Phi$ . This function maps from  $\mathbb{R}^d$  to  $\mathbb{R}^m$ , and is applied independently to each of the  $n$  points contained in  $x - B$ . The result lies in the domain  $\mathbb{R}^{n,m}$ . The embedding output is then passed through a permutation-invariant aggregation function  $F : \mathbb{R}^{n,m} \rightarrow \mathbb{R}^m$ . This ensures that the discriminator output will be independent of the ordering of points within the batch. Our setup implements the embedding  $\Phi$  using 3 1D convolutional layers with a kernel size and stride of 1 and 256, 256, and  $m$  filters. Several aggregation functions were considered, such as a mean, a sum, or a standard deviation. All performed similarly well, however a standard deviation resulted in slightly better results, leading us to use the standard deviation for this and all subsequent networks.

The result of this combined embedding and aggregation is then combined with the original data point  $x$  and fed into the discriminator network. This network is again an FCN with 7 layers, 1 output node, 256 hidden nodes, and  $d + m$  input nodes. Similar to the generator, the discriminator also features dropout layers with a 50% factor after each hidden layer. The entire discriminator, including the embedding, uses a LEAKYRELU activation function with a leakage of 0.01, except for the final layer, which uses a SIGMOID activation function. The combined generator and discriminator structure setup including the minibatch discrimination can be seen in Figure 5.3.

To further improve the discriminator training, a gradient penalty regularization term [150] is used in the discriminator loss. This gradient penalty works similarly to the one described in Section 4.3 and is intended to prevent the generator from becoming stuck due to vanishing gradients in the discriminator. Additionally, we smear the training data with a uniform noise in the range  $[-0.1, 0.1]$ . This ensures the discriminator cannot simply learn the 100 training points by heart.

Every network was implemented using PYTORCH [151]. The discriminator and generator were alternately trained using the ADAM [108] optimizer with a  $5 \times 10^{-5}$  learning rate and the parameters  $\beta_1 = 0.5$ ,  $\beta_2 = 0.9$ . This corresponds to a lower momentum contribution than what ADAM uses by default. A lowered momentum term has been found to reduce the oscillations in GAN trainings [152], as it allows the generator to more quickly change course. The setup is trained for a fixed duration of 10,000 epochs with no early stopping criterion. During this training, the learning rate is reduced to 90% of its current value every 1000 epochs. The training uses a batch size of 10 and the entire data set is shuffled after every epoch to ensure the composition of the individual batches is not constant.

This GAN is trained using the same training set that was also used in the parameter fit and subsequently used to generate a varying number of new data points. One example of this GAN-produced data is shown in the orange histogram in fig. 5.4.

We now use the quantile MSE to compare the agreement of the training data, the fit and the GAN-produced samples with the true distribution. Figure 5.5 shows the  $\sqrt{\text{MSE}}$  as a function of the number of samples generated from the GAN. We sample 100 independent versions of the training set and use each to perform one fit and train one GAN. This allows for an uncertainty

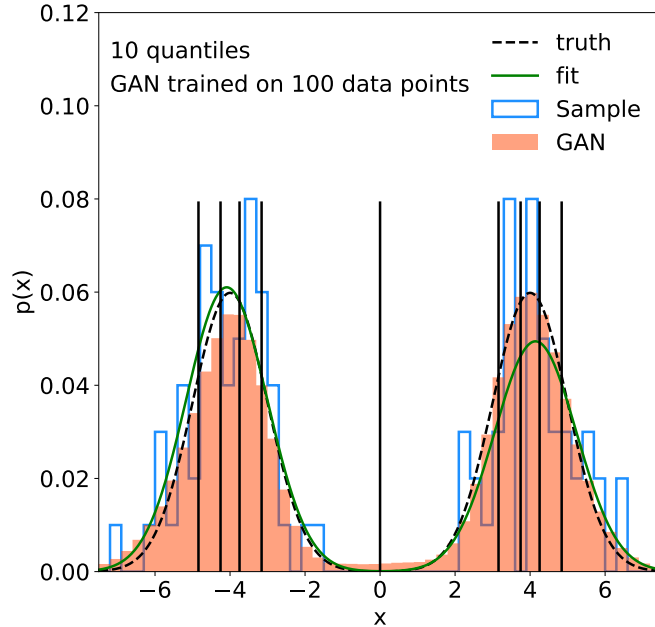


Figure 5.4: Camel back distribution. The blue and green lines show the training data and parameter fit, as in the previous plot. The orange histogram depicts data points generated using the GAN network. The black vertical lines again indicate the quantile regions. Figure adapted from [1].

estimation by calculating the standard deviation over these 100 experiments. This uncertainty is indicated by the error envelopes in the plot.

In the plot, we can see that both the training data MSE and fit MSE are flat lines. This is because the training data and fit are independent of the GAN, and their agreement with the true camel back, therefore, remains constant regardless of the number of GAN samples. Further, we can see that the fit has a better agreement than the training data alone. This is reasonable as the use of the true function in the fit adds additional information, resulting in an overall better description of the true distribution. The GAN MSE can be seen to initially drop lower the more point are sampled from the GAN. This lines up with the intuition that more samples result in a better agreement. However one can also see that this behavior does not continue indefinitely and the GAN MSE eventually saturates.

This can be understood by viewing the GAN as its own distribution  $p_g(x)$ . The more GAN samples one draws, the better the GAN distribution is described, explaining the initial improvement in agreement. However, while  $p_g(x)$  approximates the true distribution  $p(x)$ , will never perfectly match it. This difference between the  $p_g(x)$  and  $p(x)$  results in a lower bound on the MSE that cannot be reduced by further sampling from  $p_g(x)$ . However one can also see that the point where the GAN MSE levels off corresponds to a better agreement with the true distribution than what one gets from using the training data alone. This indicates that the GAN distribution  $p_g(x)$ , taken as a whole, results in a better description than the pure training samples. The

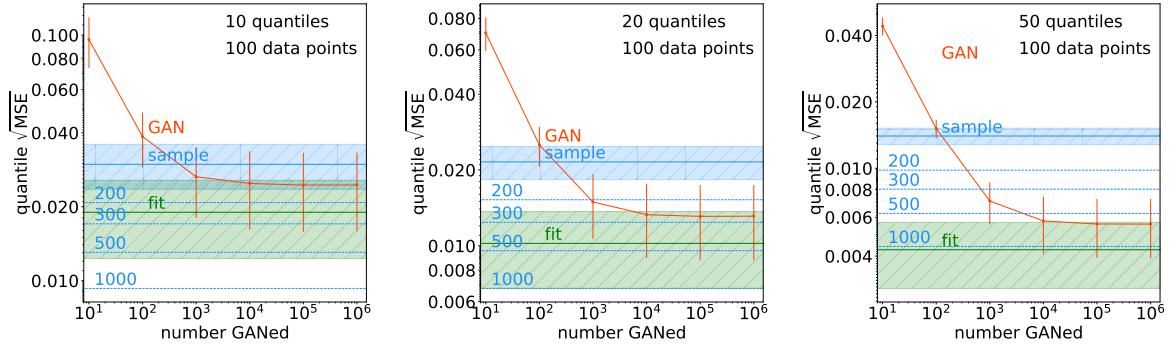


Figure 5.5: Comparison of the quantile MSE between the true camel back distribution and the training data (blue), the parameter fit (green), and the GAN (orange). Dotted blue lines correspond to larger data sets with 200, 300, 500, or 1000 points. The three panels from left to right show the results for 10, 20, and 50 quantiles.

specific improvement can be read off using the additional horizontal blue lines in Figure 5.5. These lines correspond to the agreement one obtains from a sample containing 200, 300, 500, or 1000 data points. Using these lines as reference one can estimate that the description of  $p_x$  provided by the GAN is equivalent in terms of the quantile MSE to that provided by a  $\sim 150$  point sample in the 10 quantile case. For the 20 and 50 quantiles cases, the GAN is equivalent to  $\sim 280$  and  $\sim 600$  points respectively. This allows us to define an amplification factor as the ratio between the original size of the training data and the number of points the GAN agreement is equivalent to. These factors are 1.5, 2.8, and 6 respectively.

The explanation for this amplification behavior of the GAN can be found in the generator network. Neural networks are designed to be able to approximate a wide range of functions, and therefore have few constraints on what they can describe. However, since they need to be differentiable by construction, they tend to have a smooth and continuous output. This means that when a GAN is trained on a data set, one inherently assumes that the training distribution  $p(x)$  is also smooth and continuous. This presents additional information that allows the GAN to interpolate between individual data points and thereby arrive at an overall better agreement with the true distribution than the training data alone.

Splitting the data into more quantiles results in fewer points in each individual quantile, thereby making the interpolating behavior of the GAN more noticeable for larger quantile counts. This explains why the observed amplification factor increases as one goes from 10 to 20 or 50 quantiles.

## 5.2 2-Dimensional Data

Having seen promising results for the one-dimensional toy example, we gradually expand the complexity of our data set. The first step consists of moving to a two-dimensional problem. To this end, we expand our camel back distribution to a Gaussian ring. In polar coordinates  $r$  and  $\phi$ , the probability distributions are given by

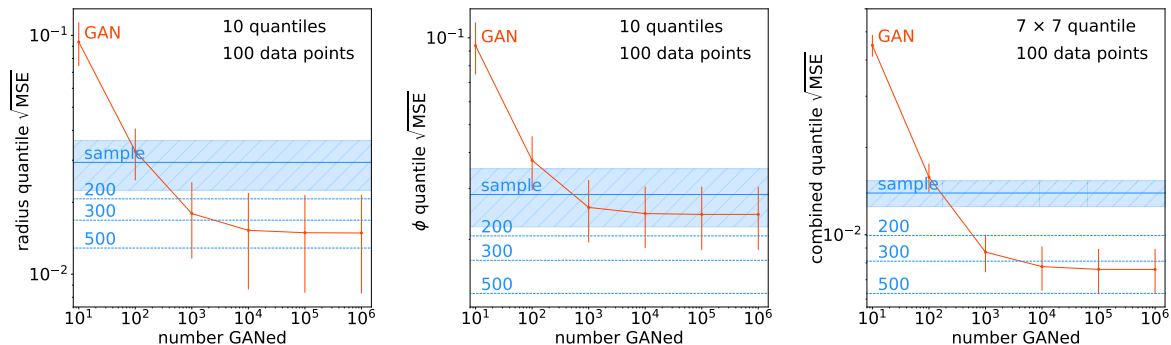


Figure 5.6: Comparison of the quantile MSE between the true Gaussian ring distribution and the training data (blue) and the GAN (orange). Dotted blue lines correspond to larger data sets with 200, 300, or 500 points. The three panels from left to right show the results for the  $r$  direction, the  $\phi$  direction, and the combination of both.

$$p(r) = \mathcal{N}_{4,1}(r) + \mathcal{N}_{-4,1}(r) \quad \text{for } r \geq 0, \quad (5.2.1)$$

$$p(\phi) = \text{const.} \quad \text{for } 0 \leq \phi < 2\pi. \quad (5.2.2)$$

The distribution in  $r$  was deliberately chosen to be symmetrical around zero to ensure the correct normalization of the Gaussian distributions.

While the ring distribution is defined in polar coordinates, the representation used to train the GAN will be in Cartesian  $x, y$  coordinates. Unlike  $r$  and  $\phi$ ,  $x$  and  $y$  are not independent and instead have strong correlations in order to form the ring distribution. Therefore we can use the Cartesian representation to investigate how well the GAN can handle correlated input variables and how this affects the amplification behavior.

To create the training set, we again sample 100 points from the Gaussian ring. The GAN setup remains largely unchanged compared to the one used in the one-dimensional experiment. The only modifications that were performed were changing the numbers of input and output nodes to accommodate the two-dimensional data set.

As a parameter fit becomes increasingly complex for larger dimensions we no longer include the fit comparison in the two-dimensional case and instead focus on the GAN and the training data.

To compare the agreement between the true distribution and GAN distribution, the data sampled from the GAN is transformed back to polar coordinates. Then the quantiles of the  $p(r)$  and  $p(\phi)$  distributions are calculated and used to determine the quantiles MSE. The results of this are shown in the left and middle panels of Figure 5.6. One can see that the GAN does a remarkable job learning  $p(r)$ , achieving an amplification factor of  $\sim 4$ . Further, while the GAN performs less well at modeling the flat  $p(\phi)$  distribution, the model still reaches an amplification factor of  $\sim 1.5$ , comparable to the 10 quantile result in one dimension.

Further, if one wants to look at the distribution as a whole, it makes sense to consider both quantile directions at once. To achieve this one can directly combine the quantiles in  $r$  and  $\phi$  direction, since  $p(r)$  and  $p(\phi)$  are independent. This results in a spider web-like structure of

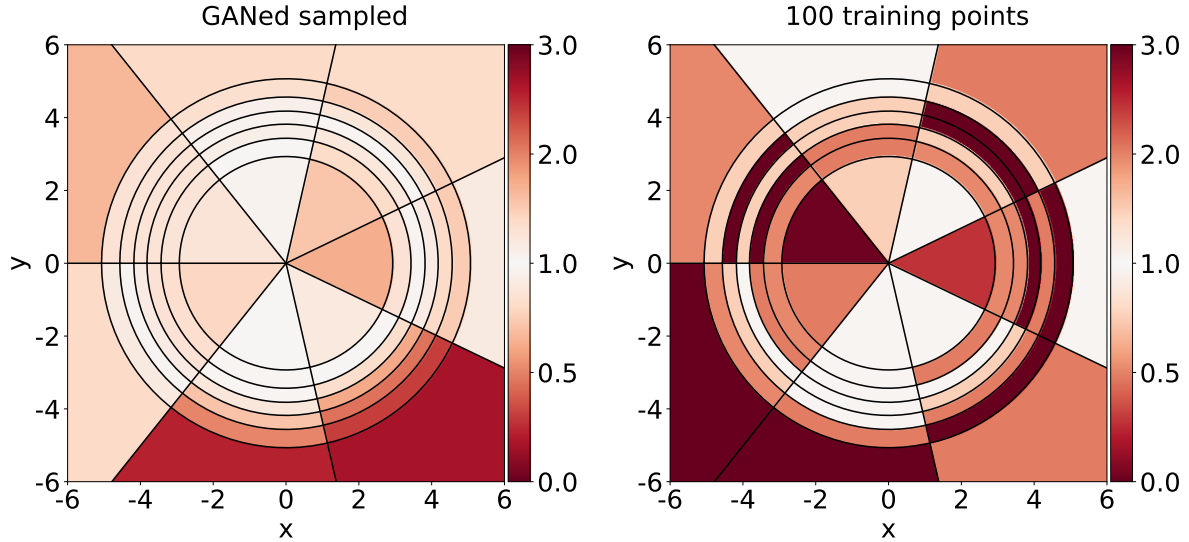


Figure 5.7: Visualizations of the combined quantile regions in  $r$  and  $\phi$  direction. Color in each region shows the deviation from the expected  $\frac{1}{N_{\text{quant}}}$  fraction, with deeper reds corresponding to a larger deviation.

quantiles shown in Figure 5.7. The figure also shows the quantile fractions of one set of 100 training points and one GAN trained on this training set. The color in each quantile region indicates the deviation from the expected quantile fraction of  $\frac{1}{N_{\text{quant}}}$ . Colors closer to white indicate a smaller deviation. Overall one can read off that the GAN achieves a smoother and more accurate description of the true distribution than the training data. This can further be quantified by calculating the MSE on this combined set of quantiles. The result of this can be seen on the right panel of Figure 5.6. Here the GAN once again has a significantly better agreement than the training data, reaching an amplification factor of  $\sim 3.5$

### 5.3 Multi-Dimensional Data

In the final proof of concept example, we further extend the number of dimensions to five. This results in a five-dimension hypersphere with a Gaussian radius. The distribution along the radius is again given by

$$p(r) = \mathcal{N}_{4,1}(r) + \mathcal{N}_{-4,1}(r) \quad \text{for } r \geq 0. \quad (5.3.1)$$

The angle distributions  $p(\phi_1)$  to  $p(\phi_4)$  are such that the vectors of the sampled points lie uniformly distributed on the surface of the five-dimensional unit sphere. The lengths of these vectors are then scaled by the Gaussian radius defined in Equation (5.3.1) to obtain the final distribution.

The increase in dimensionality also leads to an increase in the sparsity of the data, as the points are spread out over a larger volume. Therefore we increase the number of points in our

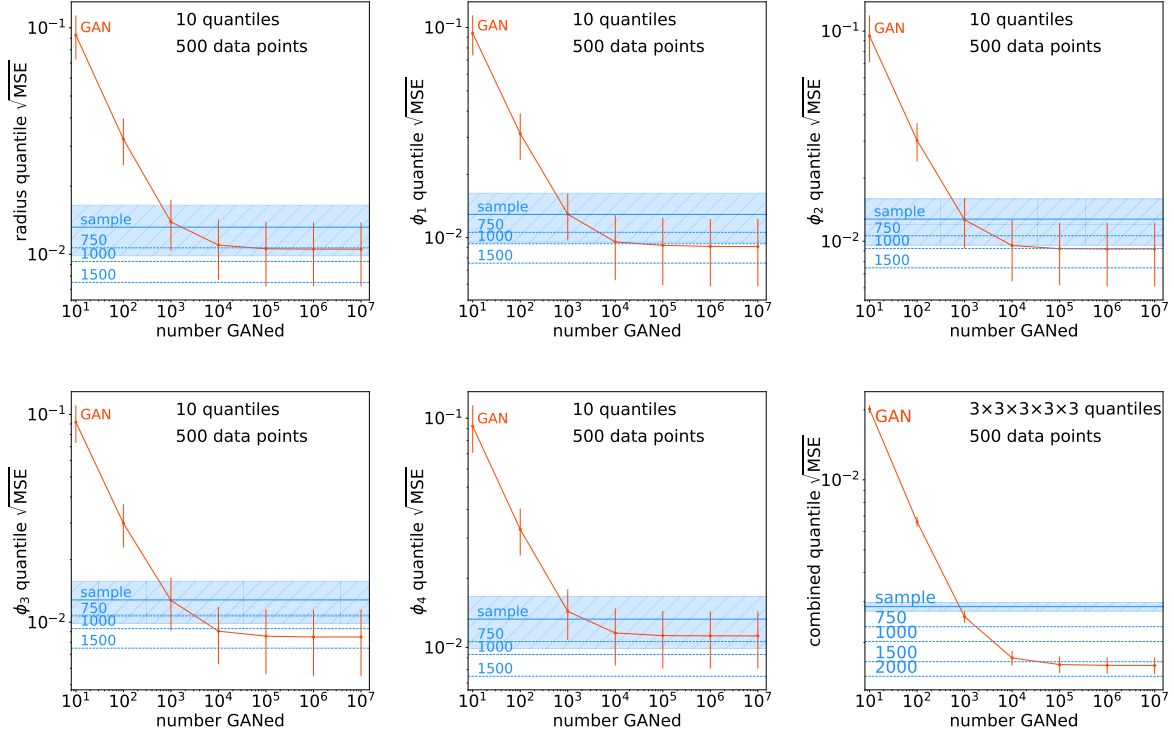


Figure 5.8: Comparison of the quantile MSE between the true five-dimensional Gaussian sphere and the training data (blue) and the GAN (orange). Dotted blue lines correspond to larger data sets with 750, 1000, 1500, or 2000 points. The top left panel shows the results for the  $r$  direction. The top center and right, and bottom left and center panels show the four angles  $\phi_1$  to  $\phi_4$ . The bottom right panel depicts the combined quantile MSE in all five directions.

training set from 100 to 500, in order to ensure the training set remains representative of the full distribution.

The five-dimensional data is then treated similarly to the previous two experiments. The GAN architecture is adapted to work with 5 input dimensions and then trained on a Cartesian representation of the data. The generated and training data is then transformed back to spherical coordinates and the agreement with the true distribution is compared using the quantile MSE. The results can be seen in Figure 5.8. The first five panels from top left to bottom right show the quantile MSE along the five directions,  $r$ ,  $\phi_1$ ,  $\phi_2$ ,  $\phi_3$ ,  $\phi_4$ . In each of the directions, we see a significant amplification, with factors ranging from 1.3 to 2.3. The worst performing direction is that of  $\phi_4$ , the angle that is uniformly distributed. This aligns with the observations in the two-dimensional Gaussian ring, where the uniform angle was more difficult to model than the radius.

The bottom right panel shows the MSE for the combined quantiles in all directions. Note, that the dimensionality of the Gaussian sphere causes the total number of combined quantiles to scale with the quantiles in each direction to the power of 5. Therefore even the lowest number of quantiles investigated, 3 in each direction, results in 243 total quantiles. For the combined MSE,

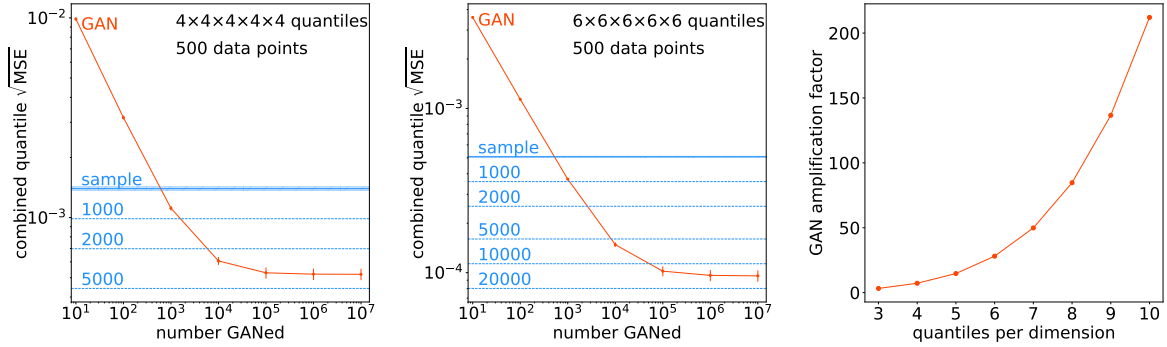


Figure 5.9: Comparison of the quantile MSE between the true five-dimensional Gaussian sphere and the training data (blue) and the GAN (orange). Dotted blue lines correspond to larger data sets. The left and center panels show the combined quantile MSE for 4 and 6 quantiles in each direction. The right panel shows the amplification factor as a function of the number of quantiles.

we can once again see how the interpolation behavior of the GAN becomes more noticeable in cases with fewer points per quantile, as GAN reaches an amplification factor of more than 3.

As MC simulations in physics have to deal with a sparsely populated high dimensional space, the question arises of what happens if the sparsity is further increased by increasing the number of quantiles. Figure 5.9 shows the effect of an increased sparsity. The first two panels plot the MSE for  $4^5 = 1024$  and  $6^5 = 7776$  quantiles, with amplification factors of  $\sim 8$  and  $\sim 30$  respectively. The final panel shows the scaling of the amplification factor as the number of quantiles increases.

## 5.4 Conclusion

The question this experiment sought to answer was how many samples can be drawn from a generative model before one becomes limited by the statistics of the original training set.

Using three model distributions we managed to show that a generative model trained on a set of data can interpolate between the data points and thereby result in an overall better description of the underlying distribution. This indicates that the model can reasonably be used to generate more points than were contained in the original training data.

We have so far only demonstrated this behavior for training sets that allow for smooth interpolation, however, this requirement is fulfilled by the majority of applications in HEP.

Therefore we can confidently say that, from a statistics view, GANs and generative models as a whole remain a viable approach for speeding up MC simulations. As such this experiment forms the cornerstone of the thesis, as it addresses one of the bigger criticisms of generative ML in HEP.





## Chapter 6

# Photon Shower Generation

The work presented in this chapter has been previously published as Reference [2] in collaboration with Erik Buhmann, Engin Eren, Frank Gaede, Gregor Kasieczka, Anatolii Korol, and Katja Krüger. Several figures and tables presented as well as the text are similar or identical to the content of this article. My contribution comprises the development and implementation of the BIB-AE network, performing the comparison studies, writing sections of the paper, handling the peer-review processes, and addressing referee comments.

In Chapter 5 we were able to demonstrate that generative models can be used to sample more points than they were trained on, thereby addressing the primary fundamental concern regarding generative FastSim. Building on these results we now move to apply generative approaches to speed up MC simulation. As discussed in Chapters 1 and 2, one of the most significant bottlenecks in MC simulation chains is calorimeter simulation. We therefore focus our efforts on the simulation of calorimeter showers in the highly granular calorimeters of the ILD [59]. We initially use photon showers, as their regular and uniform structure make them a well-understood test case.

In this chapter, we first introduce the photon shower data set in Section 6.1. Sections 6.2 and 6.3 give a detailed description of the three specific generative approaches that are used to simulate the photon showers. The performance of the generative models is evaluated by comparing the generated showers to those simulated using GEANT4 in Section 6.4, before concluding with Section 6.5.

### 6.1 Photon Data Set

The main training data consist of approximately 950k simulated photon showers within the electromagnetic calorimeter of the ILD, outlined in Section 2.6. The showers are simulated<sup>1</sup> using GEANT4 [29] version 10.4, with the QGSP\_BERT physics list. The detector was modeled using DD4HEP [153] version 1.11 within the ILCSoft [154] framework.

We define two coordinate systems;  $(x, y, z)$  for use within a calorimeter block, where  $z$  points orthogonal to the individual calorimeter layers and  $x$  and  $y$  are parallel to the layer orientation, and  $(x', y', z')$  for use within the detector barrel, where  $z'$  points along the beam direction and  $y'$  points straight upwards. The simulated photons are shot in the  $y'$  direction and once they

---

<sup>1</sup>The simulated data was provided by Engin Eren.

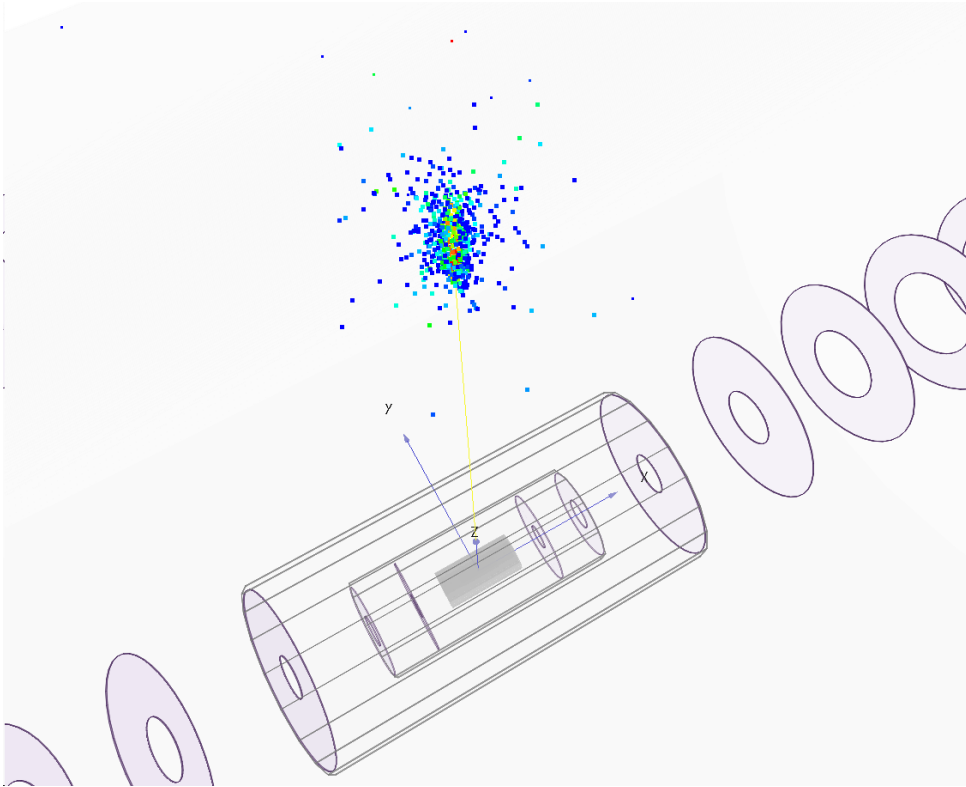


Figure 6.1: Rendering of a 60 GeV photon shower in the ILD detector. The cylindrical structure represents the ILD tracker. Figure taken from [2].

reach the calorimeter block they traverse the calorimeter along the  $z$  direction. All photons in this data set hit the detector under the same angle and are aimed to hit the center between 4 cells in the  $x - y$  plane. For this study, we consider photons with energies uniformly distributed between 10 and 100 GeV.

The resulting calorimeter hits are projected onto a regular  $30 \times 30 \times 30$  grid. This results in a tensor  $x \in \mathbb{R}^{30 \times 30 \times 30}$ , where the entries in each position indicate the energy deposited in the corresponding calorimeter cell. These tensors are what will be referred to as calorimeter images in the following.

This approach can introduce artifacts into the data, as the detector geometry itself is not perfectly regular. The most notable such artifact is the staggering of the calorimeter cells. This causes the core of the shower to appear to fluctuate between the center two cells, as can be seen in the  $y$  direction overlay of several showers shown in the center of Figure 6.2. Furthermore, the calorimeter models feature regions reserved for infrastructure, such as power supply and readout cables. Therefore, some pixel rows in the projected grid end up empty. As an artificial structure like this can be challenging for a generative model to learn, we chose to remove these empty rows by moving the closest filled rows up to take the place of the empty ones.

A separate test set was simulated in addition to the 950k training showers. This test set consists of 40k showers with a uniform energy distribution, as well as 4k showers with specific energies ranging from 20 GeV to 90 GeV in 10 GeV sized steps.

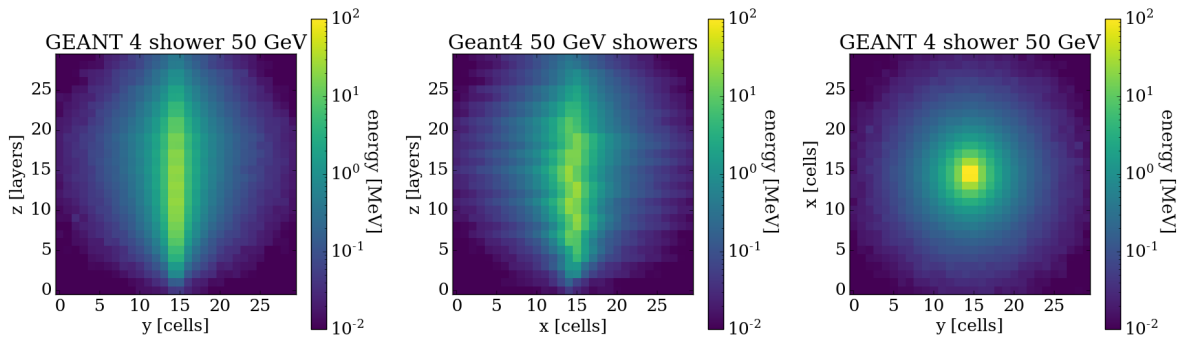


Figure 6.2: Overlays of 2000 showers induced by 50 GeV photons, projected along the (from left to right)  $x$ ,  $y$ , and  $z$  axis. The artificial structure in the  $y$  direction originates from the projection onto a regular grid. Center panel taken from [2].

Electronic noise and other disruptive factors make low energetic hits unreliable. Therefore hits below the threshold of half the energy deposited by a minimum ionizing particle ionization (0.5 MIPs) are ignored for analysis. We do not perform this cutoff to the training data, however, we do apply it to both the test data and the generated showers for the purpose of evaluating the generative performance of the models.

## 6.2 GAN and WGAN Models

We use three different generative models to learn and reproduce the photon shower data set. The first two are a GAN and a WGAN. The principles of the two methods are discussed in detail in Sections 4.2 and 4.3 respectively, therefore this section will focus on how the two architectures were specifically applied to the shower data set. A full description of the architectures and training hyperparameters is given in Appendix A.

### GAN

The GAN<sup>2</sup> architecture used in this work was broadly inspired by the CaloGAN model [48]. Our implementation uses a generator constructed out of 3D-transpose-convolutional layers and a discriminator consisting of 3D-convolutional layers, leading into a final set of dense layers. Both the generator and discriminator make use of batchnorm layers. The generator employs ReLU activation functions, in order to allow for hard cutoffs in the generated data, while the discriminator uses Leaky ReLU functions to prevent vanishing gradients in the discriminator training. The input to the generator comprises a 100-dimensional vector of noise, sampled uniformly in the range  $[-1, 1]$ .

The model is conditioned on the energy of the incident photon, so showers corresponding to a specific particle energy can be generated. To this end, the energy label is fed as an additional input to the discriminator. The generator conditioning is achieved by multiplying the noise

<sup>2</sup>The trained GAN model was provided by Anatolii Korol.

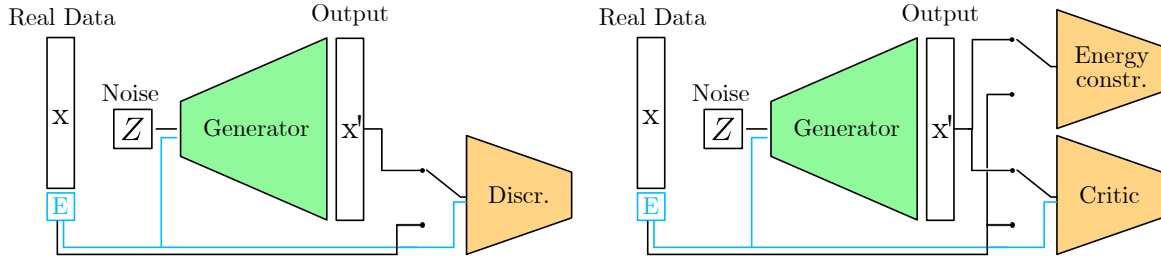


Figure 6.3: Illustration of the full photon shower GAN (left) and WGAN (right) setups. The blue lines indicate the energy conditioning. Figure adapted from [2, 3].

vector with the energy label in GeV. The blue lines in the left panel of Figure 6.3 show this energy conditioning setup.

Both networks use an ADAM optimizer [108] with a learning rate of  $2 \times 10^{-5}$  and default  $\beta$  parameters. The networks are trained in a 1 : 1 ratio, meaning that for every discriminator update there is one generator update. The total number of parameters in both models is around 3.5M.

## WGAN

Our WGAN<sup>3</sup> model is similar to the GAN model, and features a generator network built from 3D-transpose-convolutions, layernorm layers, and ReLU activations, as well as a critic consisting of 3D-convolutions, layernorm layers, and leaky ReLU activation functions. The generator input is a 100-dimensional noise vector sampled from a Normal distribution.

The model uses a gradient penalty term in order to regularize the critic to a 1-Lipschitz function.

To facilitate the conditioning on the photon energies, the energy label is fed into both the generator and discriminator as an additional input. Furthermore, our WGAN uses a constrainer network  $a$ , inspired by similar work [50], to refine this conditioning. This constrainer is trained in advance to predict the energy of an incident particle based on a given shower image. The prediction of the constrainer  $a(x)$  for given real samples  $x$  or fake samples  $x'$  is then used in an additional loss function for the generator, given by

$$L_{\text{Gen, Constrainer}} = \kappa \mathbb{E}[|(a(x') - E)^2 - (a(x) - E)^2|]. \quad (6.2.1)$$

Where  $E$  is the energy label of  $x$  and was also used as conditioning in the generation of  $x'$ . This loss can be understood as minimizing the difference between the true photon energy and the photon energy  $E$  predicted by the constrainer for a fake sample  $a(x')$ . The additional term  $(a(x) - E)^2$  accounts for cases where the constrainer prediction for a real shower differs from the true label and effectively softens the effect of the constrainer on the generator loss in such a case. The full WGAN setup including the energy constrainer is shown in the right half of Figure 6.3.

The generator and critic network use an ADAM optimizer with a learning rate of  $10^{-4}$ . This learning rate is reduced by a factor of 10 after every 50k gradient update steps.

Both the GAN and WGAN were implemented in PYTHON using the PYTORCH [151] package.

<sup>3</sup>The trained WGAN model was provided by Engin Eren.

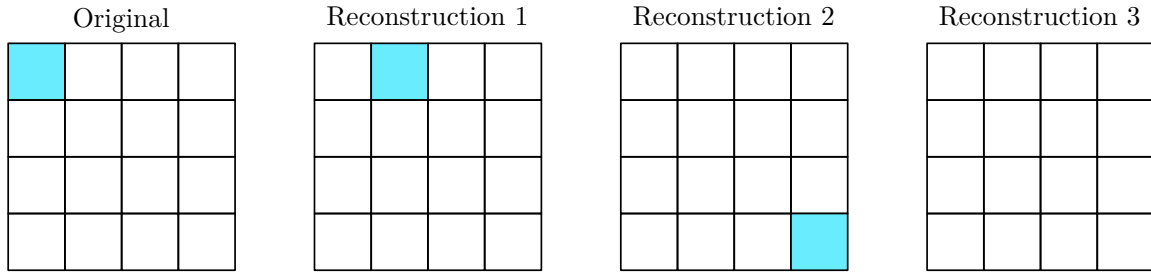


Figure 6.4: Demonstration of the MSE loss not being able to account for proximity. The MSE between *Original* and *Reconstruction 1* is identical to the MSE between *Original* and *Reconstruction 2*. Further, the MSE between *Original* and *Reconstruction 3* is lower than for *Reconstruction 1* or *Reconstruction 2*, despite containing no filled pixels.

### 6.3 BIB-AE Model

The third generative model we investigated was the BIB-AE architecture described in Section 4.7. While the practical implementation closely follows the principles outlined in that section, some modifications had to be made to accommodate for the shower data set. A complete description of details of the architecture and training hyperparameters can be found in Appendix A.

#### MSE loss on Sparse Data

Section 4.7 describes two losses designed to improve the quality of the generated data. A GAN-like adversarial network to gauge the quality of individual generated images, and an MSE loss that enforces similarity between the encoder input and decoder output.

This MSE loss is a reasonable reconstruction loss for data sets such as photographs or artwork, where every pixel is filled, however on sparse data such as our calorimeter images it leads to problematic behavior.

To explain this problem, we assume a simplified case where the input data consists of a 2-dimensional image with one filled pixel at position  $(x = 0, y = 0)$ , illustrated in the left panel of Figure 6.4. In an imperfectly reconstructed image, this one pixel can end up moved to a different position, such as  $(x = 1, y = 0)$ , or  $(x = 4, y = 4)$ , labeled reconstruction 1 and 2 in Figure 6.4. If one calculates the MSE between the original image and the two reconstruction results, then both result in the same MSE loss value, since the MSE loss has no way to account for the proximity of individual pixels. Therefore, reconstructions 1 and 2 both achieve the same MSE loss, despite reconstruction 1 being significantly closer to the original input. A further, even more problematic effect becomes apparent when one calculates the MSE loss between the input and an empty image, called reconstruction 3 in Figure 6.4. This MSE loss is lower than the ones for reconstructions 1 and 2, despite the fact that it is missing the one defining feature of the input image. Therefore the decoder network is incentivized to either perfectly reconstruct the pixel position, or to not place a pixel at all if a perfect reconstruction is impossible.

This becomes a problem when applied to the photon shower data set. Here one has a region around the shower core that features a multitude of scattered hits, as indicated on the left panel of Figure 6.5. Individually reconstructing every single one of these scattered hits is not feasible

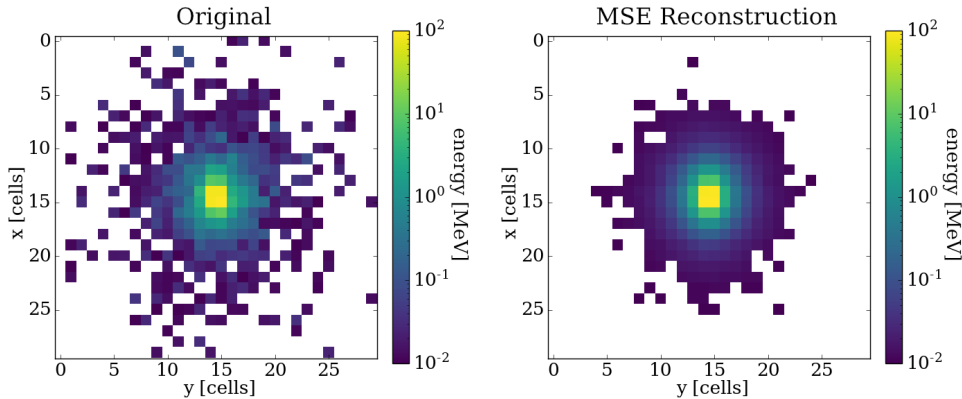


Figure 6.5: Left panel: 2d projection of a photon shower along the z-axis. Right: the same photon shower reconstructed by an MSE-trained AE.

while still following the IB approach. Therefore an encoder-decoder setup trained with an MSE loss on the photon showers will be encouraged not to place any hits in these sparse regions, as this minimizes the MSE loss. The result of this can be seen on the right panel of Figure 6.5. From a physics perspective, this is the exact opposite of what would be desired. While the exact position of the scattered hits is not of great relevance, their presence is very much important.

Therefore we chose to use a more adaptable loss function to measure the difference between input and reconstruction. This loss takes the shape of an additional *difference discriminator* network. This network is trained to distinguish between “fake” data, given by the element-wise difference between input and reconstruction  $x - D(E(x))$ , and “real” data given by a vector filled with zeros. This network forces the encoder-decoder setup to minimize the difference between input and reconstruction. However, unlike the MSE loss, the network can learn to put an unequal emphasis on different pixel positions, thereby ensuring the shower core is correctly reconstructed, while not interfering with the scattered hits around the core.

In the practical implementation, this difference discriminator is integrated into the GAN-like discriminator, resulting in one large discriminator network. The inputs to this combined network consist of  $x$  and 0 with a “real” label or  $x'$  and  $x - x'$  with a “fake” label.

## WGAN-like Critic Networks

Using a classical discriminator with the difference input outlined above would result in an immediate breakdown of the training as the discriminator will learn that any value other than 0 in the difference input corresponds to a “fake” label. Therefore one needs an approach that measures the distance between the two distributions. This is achieved by using a gradient penalty critic network instead of a discriminator. Therefore all the discriminator networks described in Section 4.7 are implemented using WGAN-like critic networks.

## Maximum Mean Discrepancy Loss

One significant difficulty in generating calorimeter showers is reproducing the precise distribution of individual hit energies in the showers, also called the visible cell energy spectrum. The

training data has a visible peak in this spectrum around the energy of 1 MIP (0.2 MeV for this calorimeter), which can be seen in the left panel of Figure 6.8. This peak plays an important role in the calibration of a calorimeter, making its correct modeling a priority. However a purely adversarial loss appears incapable of correctly replicating this feature, as can be seen from the GAN and WGAN results in the left panel of Figure 6.8.

This can be addressed through the use of an additional loss term, explicitly enforcing this feature. One such option is a Maximum Mean Discrepancy Loss (MMD) [135]. The MMD is a kernel-based method capable of directly comparing two distributions based on samples from those distributions. An MMD loss between the real cell energy distribution  $C_r$  and the generated distribution  $C_g$  is given by:

$$\text{MMD}(C_r, C_f) = \mathbb{E}[k(x, x')] + \mathbb{E}[k(y, y')] - 2\mathbb{E}[k(x, y)], \quad (6.3.1)$$

where  $x$  and  $x'$  are samples drawn from  $C_r$ ,  $y$  and  $y'$  are samples drawn from  $C_f$  and  $k$  is a positive definite kernel function. This term becomes minimal if  $C_r$  and  $C_f$  are identical. In order to evaluate the MMD loss between two sets of samples  $\{x_n\}_{n=0}^N$  and  $\{y_n\}_{n=0}^N$ , one needs to evaluate the three terms pairwise for each of the pairings  $\{x_i, x_j\}, i \neq j$ ,  $\{y_i, y_j\}, i \neq j$  and  $\{x_i, y_j\}$  respectively. Therefore the computation time scales quadratically with the number of samples  $N$ . This means simply comparing two calorimeter images, each containing  $30^3$  samples from the total cell energy distributions, would require evaluating Equation (6.3.1) approximately  $(30^3)^2$  times. This is not feasible even with significant parallelization. However, of the  $30^3 = 27,000$  cells, only around 2,000 are above the MIP cutoff and therefore relevant. Additionally, there is little sense in comparing the highly energetic parts of the cell energy spectrum to the lower regions. This allows us to speed up the MMD calculation using what we call the Sorted-Kernel-MMD.

Initially, both showers are flattened and their hit energies are sorted. Then, one takes the 2,000 most energetic hits and discards the rest. Finally, the first  $n$  hits from both flattened and sorted showers are compared using Equation (6.3.1). This comparison window is then moved by  $m$  positions and the MMD is computed on the hits in the interval ranging from  $m$  to  $m + n$ . The window continues to move until it reaches the end of the 2,000 hits. The final MMD loss is given by the sum over all intermediate calculations. This process significantly speeds up the MMD calculation, making it viable for training the BIB-AE model.

For the photon shower BIB-AE, the chosen parameters are  $n = 100$  and  $m = 50$ . The chosen  $k$  is a Gaussian kernel function given by

$$k(x, x') = e^{-\alpha(x^2 + x'^2 - 2xx')}, \quad (6.3.2)$$

where  $\alpha$  dictates the scale of structures the MMD loss is sensitive to.  $\alpha = 200$  was found to suitably resolve the peak around the 1 MIP region.

Adding this MMD loss to the BIB-AE model does in fact improve the modeling of the low energy region, however, it also introduces unphysical artifacts in the generated showers, such as superfluous hit clusters around the edges of the image.

## Post Processor

Since the inclusion of the MMD loss into the main BIB-AE network results in a worsened generation performance, we decided to offload the correct modeling of the MIP peak onto a secondary

Post Processor network. This Post Processor consists of a series of 3d-convolutional layers, with kernel size 1 and stride 1. Therefore the Post Processor can be seen as a nonlinear function applied independently to each cell. This allows the Post Processor to adjust the values of pixels but makes it so the pixels cannot be moved around, thereby preventing the introduction of unphysical artifacts through the Post Processor.

The Post Processor is only supposed to improve the cell energy spectrum by slightly adjusting individual pixel values. Therefore we want the network to be as close to the identity mapping as possible, while still correcting the spectrum. To this end, the Post Processor network is initially pre-trained with an MSE loss between its input and its output. After one epoch of pre-training, the MMD loss between the cell energy distribution of the Post Processor output and the real data is added to the MSE loss. This teaches the Post Processor to correct the mismodeled MIP peak, while otherwise performing minimal changes. The Post Processor network is trained in parallel with the main BIB-AE to ensure the Post Processors correlations are tuned to the BIB-AE network, even as the main BIB-AE changes. Furthermore, we found the Post Processor training to converge better if the BIB-AE is pre-trained for a number of epochs before the Post Processor is turned on, as the significant changes to the BIB-AE output during the first few epochs can confuse the Post Processor.

These modifications lead to the full BIB-AE model depicted in Figure 6.6. The encoder model is constructed of an initial series of 3d-convolutions, interspersed with layernorm layers, the output of which is fed into a set of fully connected layers. These map onto a 24-dimensional Gaussian latent space. The encoded latent space is concatenated with an additional 488 noise variables drawn from Normal distributions. This is done to increase the number of random variables the decoder network has access to. The decoder network uses a combination of dense layers, 3d-transpose-convolutions, 3d-convolutions, and layernorm layers to map the latent space to a  $30 \times 30 \times 30$  output. Both the encoder and the decoder use leaky ReLU activation functions.

The combined critic network is built similarly to the encoder using 3d-convolutional layers and layernorm layers, followed by a series of dense layers. The difference vector input is directly fed into the dense section of the network. The latent critic consists of a simple fully connected network. All critic networks employ leaky ReLU activations. In order to assist the latent critic, an additional MMD term was added to the loss that compares the encoded latent space to the desired Gaussian latent space.

The BIB-AE is conditioned on the photon energy by passing the energy label to the encoder, decoder, and critic networks. The final Post Processor network is also conditioned on the energy, in case the exact correction of the MIP peaks is dependent on the photon energy.

All component networks are trained using an ADAM optimizer with an initial learning rate of  $5 \times 10^{-4}$  for the encoder, decoder, and critic, and  $2 \times 10^{-3}$  for the latent critic. All learning rates are multiplied with a factor of 0.95 after every epoch, resulting in an exponentially decaying learning rate. The entire BIB-AE model was implemented using PYTORCH.

## 6.4 Results

All three generative models that were introduced in Sections 6.2 and 6.3 are trained using the training set of 950k photon showers. We now evaluate the performance of the individual models by comparing their generated showers to the test set. Unless specified otherwise, every com-



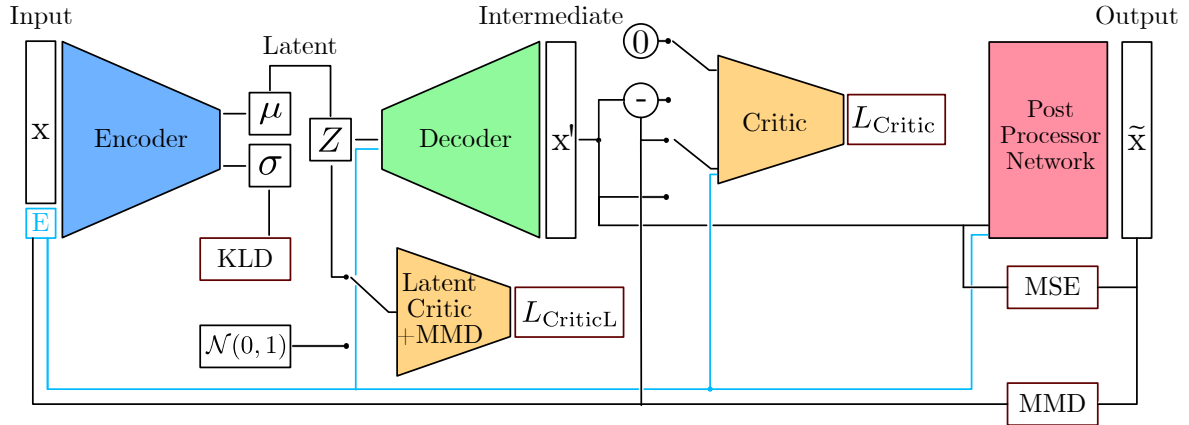


Figure 6.6: Illustration of the full photon shower BIB-AE setup with the Post Processor network. Image adapted from [2, 3].

parison in this section is done after applying a 0.5 MIP (0.1 MeV) threshold cut to both the generated and test data.

Figure 6.7 shows a visual comparison between the individual 50 GeV showers produced by the generative model and taken from the test data. From this, one can see that the generated showers look very similar to the ones produced by GEANT4, and that no generated shower shows any obvious unphysical artifacts.

### Marginal Distributions

While individual shower comparisons are helpful in identifying any glaring problems the generated showers may have, they are insufficient for our goals. In order to have a successful fast simulation approach one needs to ensure that not only the individual images are realistic, but also that the overall properties of the generated showers match those produced by GEANT4. To this end, the next step consists of comparing the physical observables of several generated showers to the same observables taken from the test data.

The first observable we consider is the cell energy spectrum. The left plot of Figure 6.8 shows this spectrum for the three generative models in the three colored lines and GEANT4 as the gray, filled histogram. The hatched region in the plot shows the part of the spectrum that is cut away by the MIP cutoff, meaning that mismatches in this region do not have an impact on the shower quality. One can see that the highly energetic hits are modeled well by all three models. In the low energy region close to the cutoff there is a visible peak in the GEANT4 spectrum, around the energy of 1 MIP. This peak is not modeled by either the GAN or the WGAN. The BIB-AE, however, manages to accurately model this peak, largely due to the dedicated Post Processing. This presents a significant success, as it demonstrates how the shortcomings of a generative model can be addressed with a specialized secondary network.

The right plot of Figure 6.8 compares the total number of pixels with an energy deposit above the cutoff threshold for photon showers caused by 20, 50, and 80 GeV photons. The GAN and WGAN can be seen to underestimate the total number of hits, while the BIB-AE model is nearly spot on. This is directly correlated to the previously discussed modeling of the cell

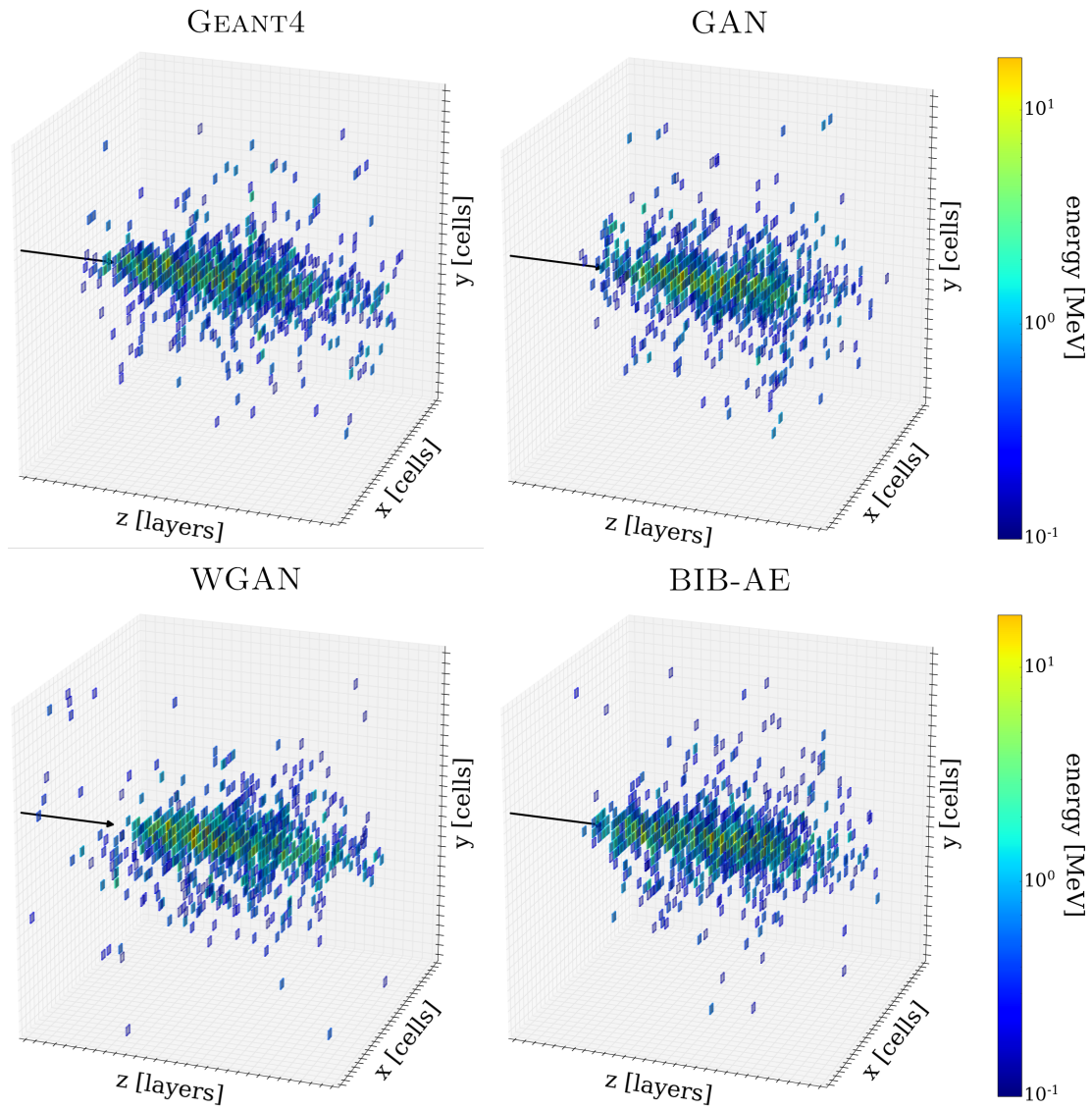


Figure 6.7: 3D renderings of individual showers generated using GEANT4 (top left), the GAN (top right), the WGAN (bottom left), and the BIB-AE (bottom right). The colors of the individual pixels indicate the deposited energy in that pixel. Figure taken from [2].

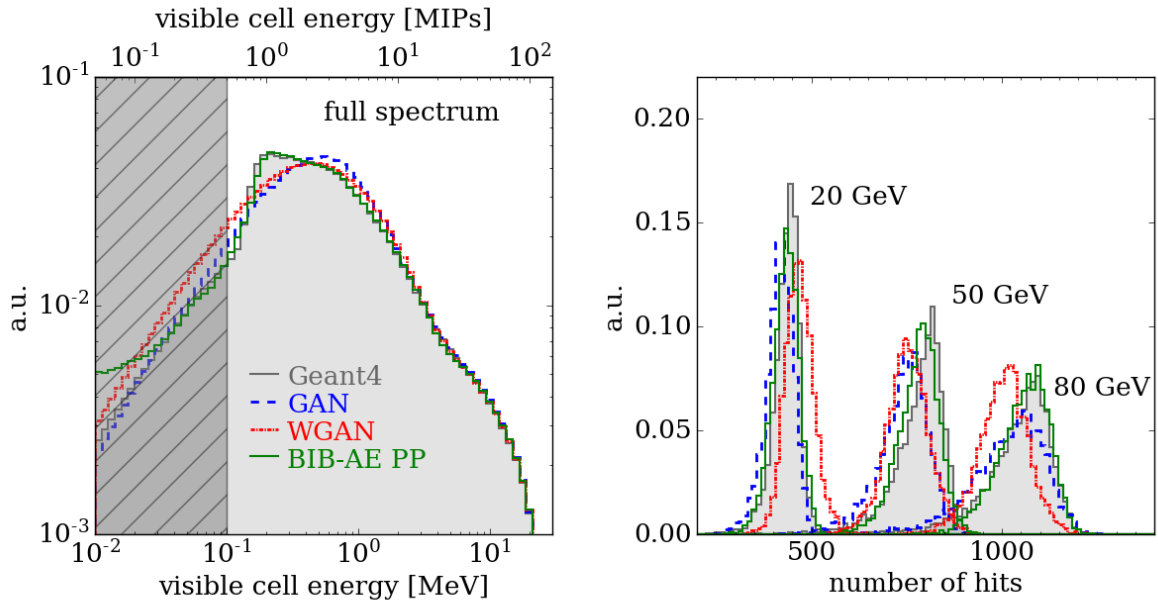


Figure 6.8: Comparison of the cell energy spectrum (left) and the number of hits in a shower for 20, 50, and 80 GeV (right). The hatched region in the cell spectrum indicates the region cut by the MIP threshold. The lines correspond to GEANT4 (filled, gray), the GAN (red, dotted), the WGAN (blue, dashed), and the BIB-AE (green, solid). Figure taken from [2].

energy spectrum. Since the exact number of points that end up above the threshold depends on the cell energy spectrum around the threshold, correctly modeling this spectrum is essential to accurately describe the total number of hits. Both GAN setups have trouble reproducing the cell energy spectrum, and therefore also struggle with the number of hits, while the BIB-AE succeeds. Beyond this, the number of hits is also an indication of how well the energy conditioning of the models is performing, as photons with higher energies should produce showers with more hits. Under this lens, one can see that the models have successfully learned a dependence between the number of hits and the photon energy.

A further evaluation of the energy conditioning can be seen in the top left panel in Figure 6.8. This plot shows the distribution of the sum over all hits above the MIP threshold in a shower, also known as the visible energy sum distribution. This is shown for 20, 50, and 80 GeV photon showers. The means of the individual peaks are well reproduced by all distributions, however both GANs show some deviations around the right flank of the distributions, where they seem to mismodel the asymmetric shape of the GEANT4 peaks. The BIB-AE appears more successful in this regard, closely reproducing both the peak widths and shapes.

Figure 6.8 additionally shows the center of gravity of the showers along the  $z$ -direction (top right), and the longitudinal and radial energy profiles (bottom left and bottom right respectively). These three observables give insight into how well the average shape of the photon showers is captured by the generative models. One can see that both energy profiles are well modeled by both the BIB-AE and the GANs, with only the WGAN showing slight deviations around

the outer layers. Even the comb-like structure in the longitudinal profile, caused by the exact orientation of the active detector layers, is learned correctly. In the center of gravity comparison, the BIB-AE can be seen to underestimate the width of the distribution, while both the GAN and WGAN demonstrate a significantly better agreement with GEANT4.

A vital property of a calorimeter is how it responds to different particle energies. In order to see in detail how well this is reproduced by the generative models, we use them to sample showers produced by photons with energies between 20 GeV and 90 GeV in 10 GeV steps. The showers are then compared to the test set containing GEANT4 showers for the same energies. To this end, the showers are summed to obtain the visible energy sum distributions (partially shown in the top left of Figure 6.9) and then the means and widths of the individual distributions are calculated. We define the mean of the distributions as the mean of the central 90% interval, also called the  $\mu_{90}$ . Similarly, the width is defined as the standard deviation of this interval, called  $\sigma_{90}$ .

The results of this are plotted in Figure 6.10. On the left one can see how well  $\mu_{90}$  is reproduced. The upper section of the plots shows a high level of agreement between GEANT4 and the generative models. The ratio plot in the lower section confirms this, showing deviations in the range of only 2% for the GAN and BIB-AE. The  $\frac{\sigma_{90}}{\mu_{90}}$  comparison shown on the right reveals that the relative width<sup>4</sup>, however, is not as well modeled. Both GANs overestimate the width by about 25%. The BIB-AE performs significantly better, with its  $\mu_{90}$  centered around the GEANT4 value, however, it still has deviations of up to 15%. This is in line with the top left panel of Figure 6.9, where one could see the BIB-AE more closely describing the visible energy peaks.

## Correlations

The final criteria we apply to judge the quality of the generative shower is whether their internal correlations are in line with those expected from the GEANT4 data. To this end, we calculate the pair-wise Pearson correlations between a select set of observables, inspired by Reference [155]. These observables are: The first moments along the three directions, labeled  $m_{1,x}$ ,  $m_{1,y}$ ,  $m_{1,z}$ , the second moments along the three directions, labeled  $m_{2,x}$ ,  $m_{2,y}$ ,  $m_{2,z}$ , the visible energy sum  $E_{\text{vis}}$ , the incident photon energy  $E_{\text{inc}}$ , and the number of hits above the cutoff  $n_{\text{hit}}$ . The final three observables are ratios between the energy deposited in the first third of the calorimeter and the total visible energy  $E_1/E_{\text{vis}}$ , the ratio between the second third and the total  $E_2/E_{\text{vis}}$ , and the ratio between the final third and the total  $E_3/E_{\text{vis}}$ .

The correlation coefficients form a correlation matrix, which is calculated for GEANT4 and the three generative models. Figure 6.11 shows the correlation matrix for GEANT4 in the top left. In order to easily compare the correlation matrices, we calculate the element-wise difference between the GEANT4 matrix and the generative matrices. These difference matrices are shown in the top right and bottom of Figure 6.11, the closer to zero the differences, the better the agreement. One can see that the GAN agrees very closely with GEANT4, only showing some small deviations of 0.2 for the second moments. The WGAN performs significantly worse, showing rather large deviations in multiple correlation coefficients. The BIB-AE performance lies between the performances of the two GANs, having smaller deviations than the WGAN, but still performing worse than the GAN.

<sup>4</sup>Note that while this quantity has similar properties to the resolution of the calorimeter, its numerical values do not represent the resolution

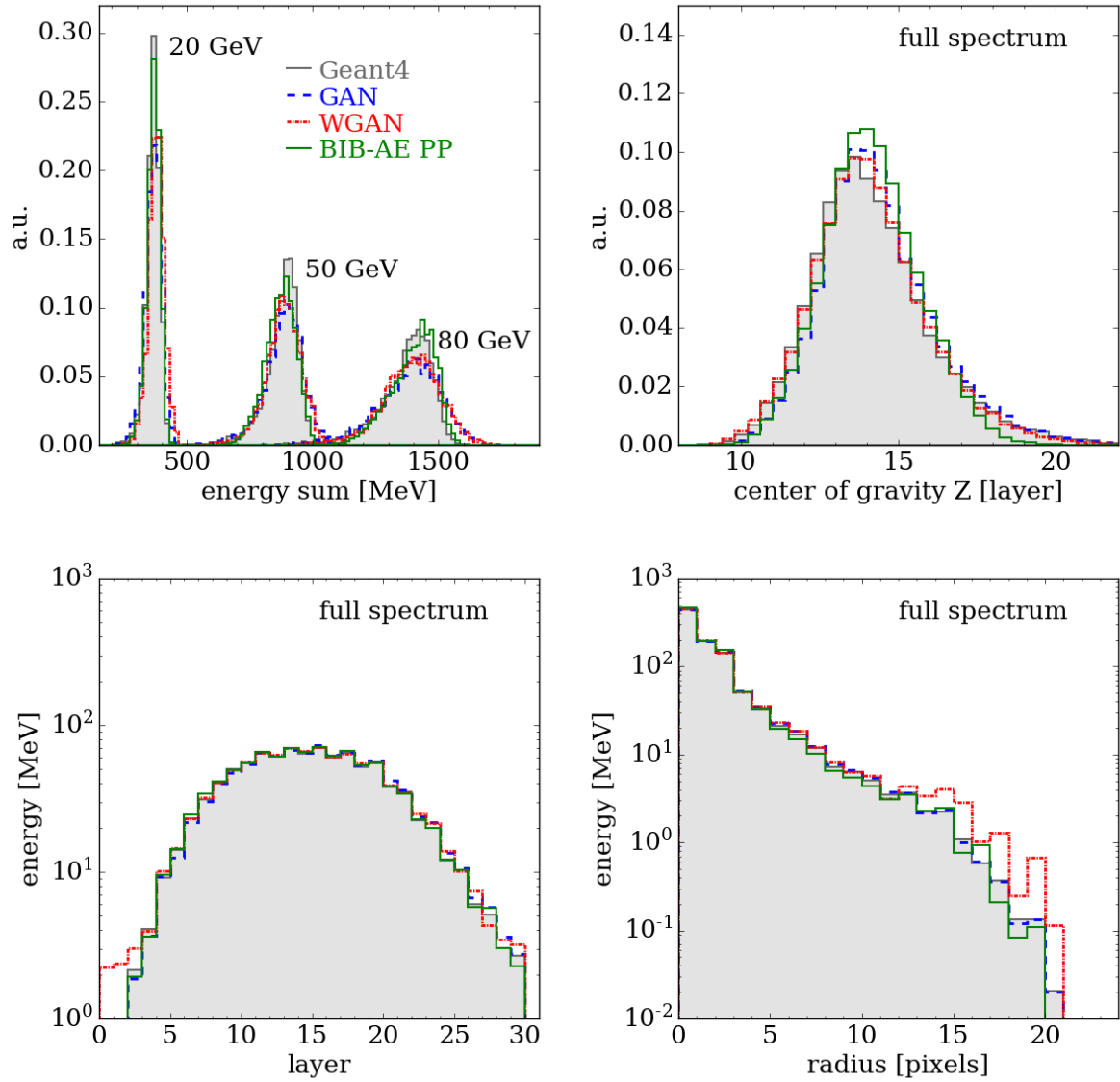


Figure 6.9: Comparison of the total visible shower energy for 20, 50, and 80 GeV (top, left), the center of gravity in the z-direction (top, right), the longitudinal energy profile (bottom, left), and the radial energy profile (bottom, right). The lines correspond to GEANT4 (filled, gray), the GAN (red, dotted), the WGAN (blue, dashed), and the BIB-AE (green, solid). Figure taken from [2].

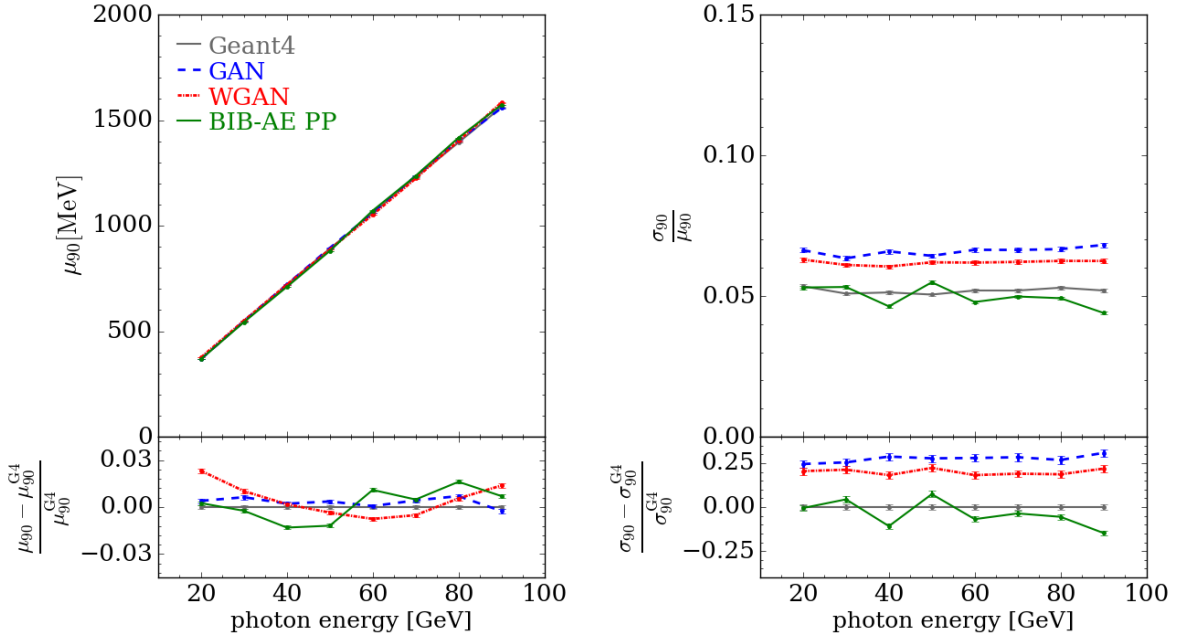


Figure 6.10: Comparison of  $\mu_{90}$  (left) and  $\frac{\sigma_{90}}{\mu_{90}}$  (right) of the visible energy sum distributions for single photon energies ranging from 20 GeV to 90 GeV. Lower plot sections show the ratios between GEANT4 and the generative models. The lines correspond to GEANT4 (filled, solid), the GAN (red, dotted), the WGAN (blue, dashed), and the BIB-AE (green, solid). Figure taken from [2].

A more detailed examination of the correlations between the visible energy and the number of hits is shown in the top row of scatter plots in Figure 6.12. The GEANT4 test data shows an approximately linear correlation, with a slight bend. The GAN manages to replicate this shape very well, as can be seen in the second panel. The WGAN on the other hand does not learn this curved shape and instead produces an almost exactly linear correlation. The BIB-AE appears to correctly learn the overall shape, however, the scatter distribution is significantly sharper, as can be seen from the higher number of points in the peaks. This is in agreement with the previous ranking of the three approaches in terms of their produced correlations.

The lower row of Figure 6.12 shows the correlations between the visible energy and the center of gravity along the z-axis. Here one can see a good agreement between GEANT4 and the GANs, however, the BIB-AE once again displays a distribution of scatter points that is too sharp. A comparison with the upper right panel of Figure 6.9 shows that the BIB-AE does model the center of gravity distribution too sharply, explaining this observation in the scatter plot.

## Computational Timings

The main goal of using generative models for shower simulation is to speed up simulation times compared to classical MC simulations. Therefore the speedup factor achieved by the generative

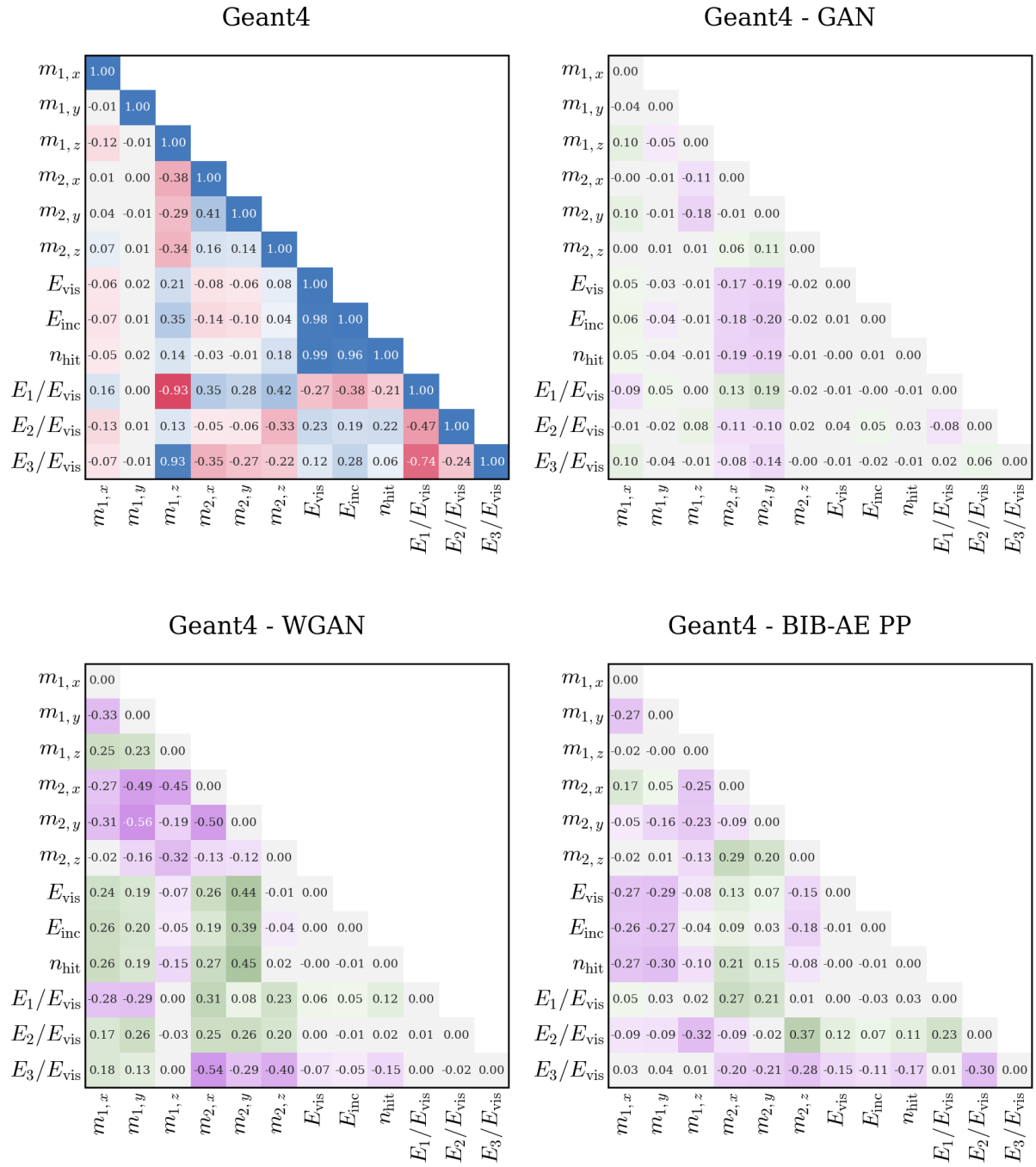


Figure 6.11: Pairwise Pearson correlation coefficients between various observables are outlined in the text. For GEANT4 (top left) the full correlations are shown, for the generative models we show the difference between the generative and GEANT4 correlations. Color scales of the difference matrices are identical. Figure taken from [2].

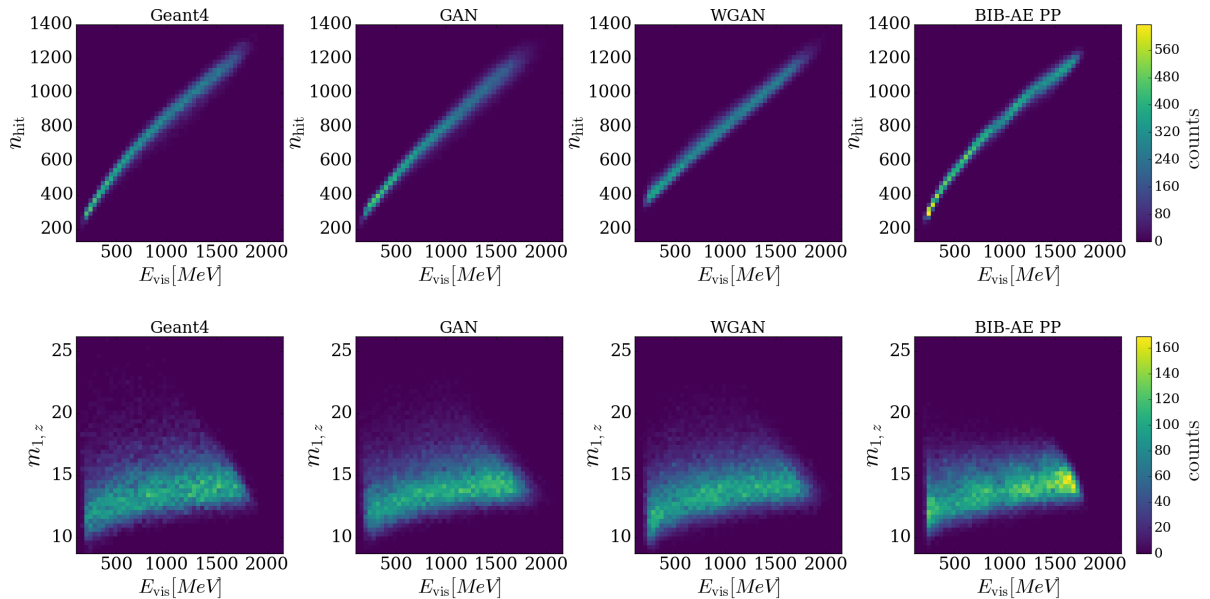


Figure 6.12: Scatter plots demonstrating the precise correlations between the visible energy sum and the number of hits (top) and between the visible energy sum and the first moment (i.e. the center of gravity) along the z-axis (bottom). Figure taken from [2].

approaches is a vital performance metric. Table 6.1 compares the time to generate individual photon showers between GEANT4, the WGAN, and the BIB-AE<sup>5</sup>. The GAN was omitted for the sake of readability, as its generation times are nearly identical to those of the WGAN.

We differentiate between two cases. On the one hand, we look at the simulation of showers caused by 15 GeV photons, while on the other hand, we simulate showers for photons with energies ranging from 10 GeV to 100 GeV. Every shower particle is modeled individually in GEANT4, leading to a higher simulation time for particles with high energies. Therefore the average time per shower for GEANT4 is almost three times higher for the energy range than for the 15 GeV photons. This energy dependence is not present in the generative approaches, as they always perform the same series of computational operations regardless of the specified particle energy. As both cases are relevant for different applications we include both in the comparison.

We further split the timing evaluation between timings performed on a standard processor (CPU) and timings performed in a Graphical Processing Unit (GPU). Running the generative models on a GPU optimized for ML operations will naturally result in faster generation times than running the models on a CPU, however as there is no GPU GEANT4 version, CPU timings are needed to allow for a fair comparison. Nevertheless, the GPU times serve as an upper benchmark, demonstrating the speedup that would be possible if the appropriate hardware is used.

When run on a CPU the WGAN model offers a speedup factor of 25 $\times$  compared to GEANT4 for 15 GeV showers, and a factor of up to 70 $\times$  for 10-100 GeV. The BIB-AE displays a speedup of 3 $\times$  and 10 $\times$  for 15 and 10-100 GeV, respectively. This is quite promising, as these factors can be achieved on the same hardware currently used to run GEANT4. Furthermore, it allows

<sup>5</sup>Generative Model and GEANT4 timings were provided by Engin Eren.



t

Table 6.1: Comparison of the per shower simulation time between GEANT4, the WGAN, and the BIB-AE. Timings for the GAN models are omitted as they are nearly identical to the WGAN. All models were evaluated on a single core of a Intel<sup>®</sup> Xeon<sup>®</sup> CPU E5-2640 v4 (CPU). Additionally the generative models were evaluated on a NVIDIA<sup>®</sup> V100 with 32 GB of memory (GPU). The generative models were evaluated using the batch sizes that resulted in the fastest evaluation. The uncertainties correspond to the standard deviation over 25 runs. Table adapted from [2].

Hardware	Simulator	15 GeV [ms]	Speed-up	10-100 GeV [ms]	Speed-up
CPU	GEANT4	1445 ± 19	×1	4082 ± 170	×1
	WGAN	57.99 ± 0.97	×25	57.99 ± 0.18	×70
	BIB-AE	419.64 ± 0.07	×3	418.04 ± 0.20	×10
GPU	WGAN	3.24 ± 0.01	×446	3.25 ± 0.01	×1256
	BIB-AE	1.42 ± 0.01	×1017	1.42 ± 0.01	×2874

for a trade-off between speed and precision, with the BIB-AE having better agreements in the marginal distributions, and the WGAN offering a better speedup factor.

On the GPU the relative ranking switches and the BIB-AE starts to reach a speedup factor of 1000× for 15 GeV showers, and almost 3000× for 10-100 GeV range. The WGAN falls somewhat behind, achieving speedup factors of around 500× and 1250×. Nonetheless, both models offer a staggering speedup when run on a GPU. This increase in speed has to, however, be balanced against the increased procurement cost of GPU compared to CPUs.

## 6.5 Conclusion

We investigated the use of three different generative models, a GAN, a WGAN, and a novel BIB-AE, to simulate photon showers in a highly granular calorimeter. In doing so we were able to show the ability of the GAN, WGAN, and BIB-AE to closely reproduce the shower features of the classical simulation software GEANT4.

Notably, we demonstrated that the BIB-AE setup, in combination with a Post Processor network, is capable of correctly modeling specific shower features such as the lower part of the cell energy spectrum or the number of hits. This makes the BIB-AE the first generative setup to accurately reproduce these features for showers in a highly granular calorimeter. This correct description of the cell energy spectrum further enables the BIB-AE to more closely reproduce the per-shower hit multiplicity of GEANT4.

All examined generative models provide a significant speedup compared to GEANT4 when evaluated on identical CPU hardware, and can achieve an even greater increase in simulation speed when used in dedicated ML hardware, such as a GPU.

These results represent an encouraging first step and show that a generative model can reproduce showers in a highly granular calorimeter, while requiring less computational resources

per shower than classical simulation methods. However, additional research may be required to further improve the precision of the model before it can be used in a real application.

Further, we were able to show the adaptability of generative networks, as hard-to-model features can be directly targeted through the use of additional dedicated networks. Such post processing approaches may also prove useful in other generative applications where high precision is required.

## Chapter 7

# Pion Shower Generation

The work presented in this chapter has been previously published as Reference [3] in collaboration with Erik Buhmann, Engin Eren, Frank Gaede, Daniel Hundhausen, Gregor Kasieczka, William Korcari, Katja Krüger, Peter McKeown, and Lennart Rustige. Several figures and tables presented as well as the text are similar or identical to the content of this article. My contribution to the publication comprised implementation and optimization of the BIB-AE model and its improvements, performing the comparison studies, writing sections of the paper, handling the peer-review processes, and addressing referee comments.

In Chapter 6 we demonstrated the use of generative ML models for photon shower simulation. However, in order to accelerate calorimeter simulation, it is not sufficient to model only one type of particle. In fact, all common particles that cause a calorimeter shower, such as photons, electrons, pions, or kaons, need to be taken into account. It can be argued that electron and photon showers are similar enough that the results of the photon shower generation can easily be transferred to electrons. However, as discussed in Chapter 2, hadronic showers produced by pions or kaons are significantly and structurally different from photon showers. Specifically, hadronic showers feature fewer overall interactions compared to electromagnetic showers. This means that, unlike the largely uniform electromagnetic showers, two hadronic showers can have vastly different shapes. A generative model aiming to reproduce hadronic showers will have to learn all of these variations, leading to a significantly more challenging task.

Therefore, it is vital to demonstrate that our fast simulation approach is also capable of simulating hadronic showers. We place our focus on charged pion showers, as we assume any result obtained for charged pion showers should be generalizable for all hadronic showers.

In this Chapter, we initially introduce the pion shower data set in Section 7.1. Section 7.2 and 7.3 describe the modification WGAN and BIB-AE used to model the pion showers, respectively. We present the results obtained from these models in Section 7.4 and conclude in Section 7.5.

### 7.1 Pion Data Set

To build the pion shower training set, approximately 500k positively charged pion showers were simulated<sup>1</sup> using version 10.4 of GEANT4 [29]. The detector model used in the simulation was the Analogue Hadronic Calorimeter (AHCAL) of the ILD, described in more detail in Section 2.6.

---

<sup>1</sup>The simulated data was provided by Engin Eren.

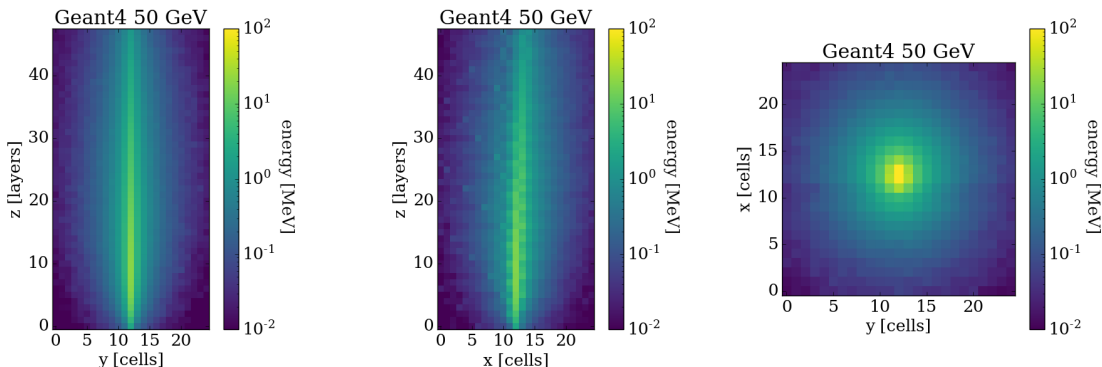


Figure 7.1: Overlays of 4000 showers belonging to 50 GeV pions, projected along the x-axis (left), y-axis (center), and z-axis (right). The slight bend in the shower core position in the center image is due to the magnetic field of the detector.

Similar to the photon data set, DD4HEP version 1.11 and ILCSoft were used as the framework for the shower simulation.

We use the same two coordinate systems  $(x, y, z)$  and  $(x', y', z')$  as in Chapter 6, within the calorimeter block and the detector barrel respectively. The  $z$  axis is defined to point along the depth of the calorimeter layers, and  $z'$  points along the beam direction.

The pions used in the shower simulation are initially placed at  $(x' = 3 \text{ cm}, y' = 100 \text{ cm}, z' = 100 \text{ cm})$  and propagate into the detector along the  $y'$  axis. The starting point was chosen such that the pions hit the AHCAL approximately perpendicularly after being deflected by the magnetic field of the detector. The ECAL that is normally in place before the AHCAL was removed from the detector simulation for the generation of the data set. This was done to ensure the pions do not shower prematurely in the ECAL, as this would dilute the training data. All pions are simulated with this starting position and direction. The pion energy is uniformly distributed between 10 and 100 GeV.

The energy deposits in the calorimeter cells are projected into a  $25 \times 25 \times 48$  regular grid. The grid size along the  $z$ -direction corresponds to the number of layers in the AHCAL. The  $x$  and  $y$  sizes were chosen as a compromise between reducing the sparsity of the data set and containing as much of the shower as possible. The choice of 25 pixels results in an average of 1.3% of pixels being filled and allows for containment of, on average, 96% of the energy of a 40 GeV pion shower. Overlays of the resulting calorimeter images are shown in Figure 7.1.

Similar to the treatment of the ECAL showers, we correct for any artifacts caused by the projection of the irregular calorimeter cells onto the regular grid. Additionally, there is a subset of pions that will not induce an actual shower in the calorimeter but will instead simply pass through, leaving a single ionization trail. These events are trivial to simulate classically, however, due to their low number of active pixels and drastically different shape compared to the bulk of the showers they present a significant challenge for a generative model. Therefore we remove these types of showers from the data set, by requiring all showers in the training set to have at least 70 hits with energies above the 0.5 MIP threshold, with 1 MIP  $\approx 0.5 \text{ MeV}$  for the AHCAL.

As the magnetic field of the detector extends into the calorimeter, the overall path of the pions has a slight bend, as can be seen in the center panel of Figure 7.1. This introduces a slight asymmetry in the x-y plane of the shower.

In addition to the 500k pion showers in the training set, we generated two test data sets. The first consists of 49k pion showers with energies uniformly distributed between 10 and 100 GeV. The second set contains pion showers at fixed energies, running from 20 to 90 GeV in 10 GeV steps. For each energy step, 8k showers were simulated.

## 7.2 WGAN Model

The WGAN<sup>2</sup> used for the pion shower generation is a direct successor of the WGAN used for photon shower simulation in Section 6.2. Therefore this description will not cover the complete model, but instead, focus on the improvements made to the model setup in order to accommodate the more complex pion showers. Additional details about the pion WGAN architecture and its hyperparameters can also be found in Appendix A.

### ResNet Critic

One major change to the model was switching the purely convolutional structure of the critic to a residual architecture [156]. The skip connections featured in this architecture reduce the risk of vanishing gradients in the critic training. This enables the use of a larger critic network with increased expressiveness.

### Fully Connected Constrainer

The second modification concerns the energy constrainer network tasked with predicting the particle energies of the generated showers. The constrainer used for the photon WGAN used a convolutional structure. However, projecting the showers from the irregular calorimeter into the regular grid structure used to train the model introduces patterns into the shower images. These patterns break the underlying assumption of translational invariance made by convolutional layers. Therefore the convolutional layers of the constrainer were replaced with a fully connected network.

## 7.3 BIB-AE Model

The pion shower BIB-AE maintains the main component of the photon generation version, such as the encoder-decoder principle, the combined difference critic, and the three latent regularization terms. However, it also features several upgrades required to improve its performance on the more challenging data set. The exact architecture of the pion BIB-AE the hyperparameters used in the training is described in Appendix A.

For the training of the BIB-AE we apply an additional cutoff to the data, reducing every pixel value below 0.1 MIP to zero.

### Complex Latent Space Sampling

The hyperparameters of the BIB-AE setup enable us to tune the amount of information contained in the BIB-AE latent space. However, the choice between a high-information latent space or a

---

<sup>2</sup>The trained WGAN model was provided by Engin Eren.

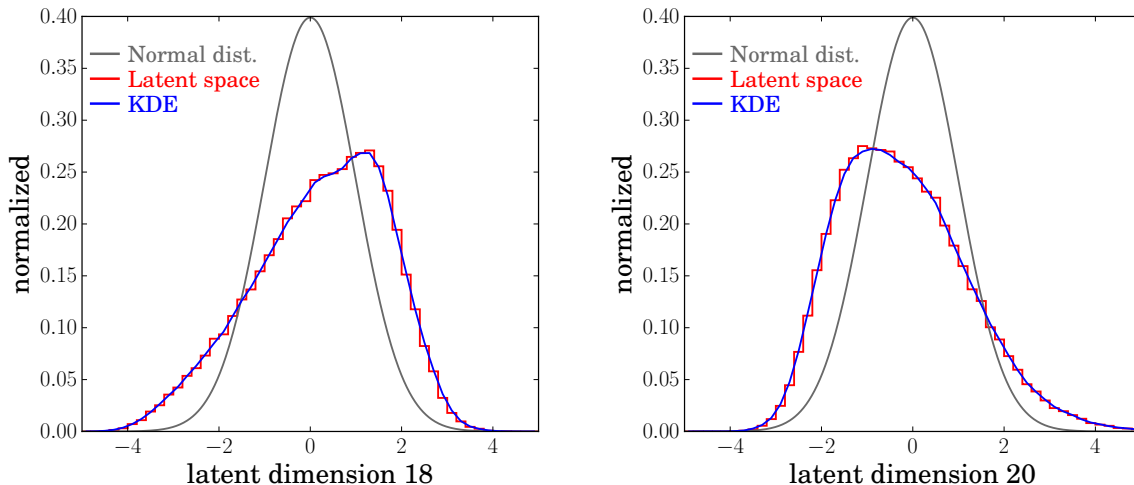


Figure 7.2: Selected latent space dimensions of the pion shower BIB-AE model. The red histogram shows the latent space produced by the encoder. The gray line shows the target latent space. The blue line shows how the KDE can learn the latent space distribution.

low-information latent space is fundamentally a trade-off. On the one hand, it is easier for the decoder to map a high-information latent space back to the data, as a large part of the information required to generate realistic samples is already provided in the latent space. Therefore, a high-information space is desirable, as it results in high-quality generated samples. On the other hand, one needs to consider that the more information is contained in the latent space, the larger its deviations from the target Normal distribution. This is problematic during sampling, as a decoder trained on a non-Normal latent space will produce worse samples if applied to a Normal distributed noise vector. The two requirements could be balanced for the relatively uniform photon showers, however, for the complex hadronic showers, the latent space shows significant deviations from a Normal distribution. For comparison, select latent space dimensions with large deviations are shown in Figure 7.2. This results in the generated shower being significantly different from the GEANT4 showers, as demonstrated in Figure 7.5.

One possible solution would be to increase the latent space regularization even further, however as outlined above this would also result in decreased generation quality. Our solution, therefore, involves a change to the noise sampling rather than the network itself. While the BIB-AE latent space deviates strongly from its target, it is still a relatively simple and low-dimensional distribution. Therefore we can fit a less complex density estimator to the latent space and sample from this estimator during generation. This principle is similar to the Buffer-VAE setup introduced in Reference [157] and its applicability to a BIB-AE setup has been demonstrated in Reference [6].

Due to its ease of use a Kernel Density Estimation (KDE) [158] was chosen to perform the latent space density estimation. A KDE can be constructed from a set of samples  $\{x_n\}_{n=1}^N$  containing  $N$  total points using

$$f_{\text{KDE}} = \frac{1}{Nh} \sum_{n=0}^N K\left(\frac{x - x_n}{h}\right), \quad (7.3.1)$$

where  $K$  is a kernel function and  $h$  is a scalable parameter dictating the smoothness of the distribution, called the bandwidth. We found a bandwidth of  $h = 0.1$  to yield the best overall shower generation results. Further, a Gaussian kernel function was chosen, as the BIB-AE latent space is per definition a combination of Gaussians. Applying the KDE to the latent space can be understood as placing a Gaussian distribution at every point in the latent space. The aggregate distribution of these Gaussians is then sampled to generate new points. The blue lines in Figure 7.2 demonstrate how well the KDE can approximate the latent space.

The complete workflow for the KDE latent sampling consists of training the BIB-AE, then using the trained encoder to encode 100k showers into latent samples. These latent samples are then used to generate the KDE, which is in turn used during the shower generation. In order to build and sample from the KDE, we made use of the KDE implementation in `SCIPY` [159].

One challenge of using the KDE is that the exact latent space shape tends to be correlated with the incident particle energy label. However, conditioning the KDE on the energy label is highly non-trivial. Therefore this problem is circumvented by including the energy label in the KDE, allowing the sampling of latent space and energy labels while keeping the correlations intact. In order to generate showers for specific incident energies, we employ a rejection sampling approach, where the KDE is repeatedly sampled until an energy label sufficiently close to the target is drawn. Due to the fast sampling speed of the KDE approach, this results in a negligible change to the overall shower generation times.

### Minibatch Discrimination

Minibatch discrimination [147] is a modification to adversarial training processes, where the adversarial network receives not only independent information about the individual input points, but also information about the composition of the entire input batch. This allows the adversarial network to more easily catch cases where the individual, generated images appear realistic, but their overall composition differs from the training data. Since the increased variety of shower shapes in the pion data set is a significant concern, we chose to introduce a minibatch discrimination setup to the main BIB-AE critic network.

To this end, the critic is modified to calculate the sum and standard deviation of each shower in an input batch. Then the pairwise difference between the sums and between the standard deviations is calculated, resulting in a set of difference matrices that describe the fluctuations within one batch. These matrices are run through an embedding network and finally fed into the main critic. Since the pixel values of the calorimeter images cover several orders of magnitude, this operation is performed both directly in the input showers and again on a log-scaled version of the input.

### Additional Resetting Critic Networks

During the critic training, it is vital to provide the critic network with sufficient examples for it to learn the differences between the training data and generated data. For the pion data set this is complicated by the fact that the individual showers are exceedingly sparse, resulting in only an

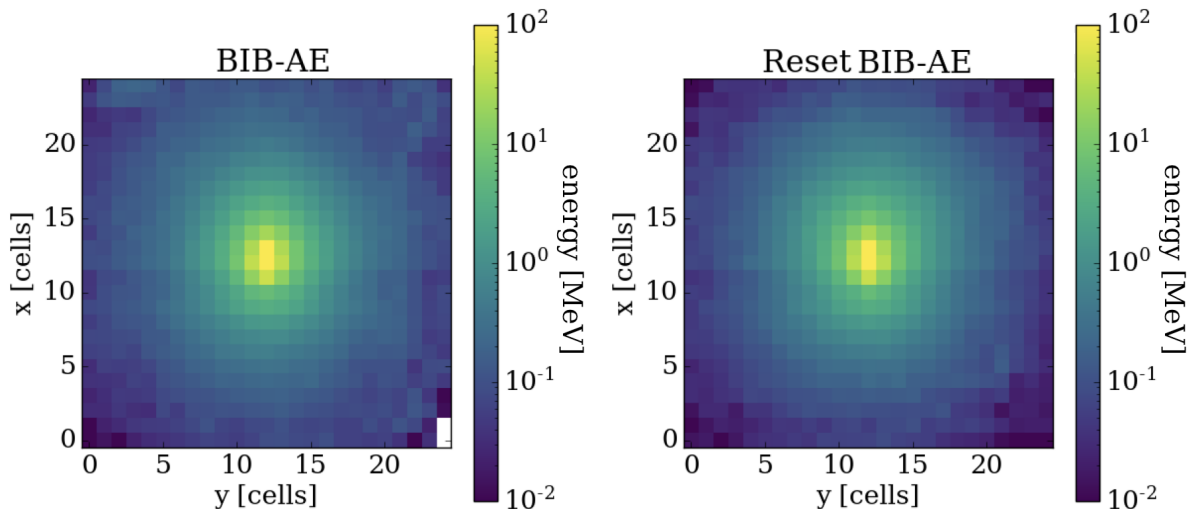


Figure 7.3: Overlays of 4000 showers belonging to 50 GeV pions, projected along the z-axis. The color scale shows the average energy in a pixel. The left panel shows the standard BIB-AE, and the right panel shows the BIB-AE with the additional resetting critic, note the artifacts on the border regions of the standard BIB-AE image are not present in the reset BIB-AE.

average of 1.3% of all 30,000 pixels having values above the MIP threshold. This means that the critic can only learn about a small number of pixels per iteration. Further, the pixel positions are not uniformly distributed but instead centered around the core of the shower. Therefore the critic will have severe difficulty learning the shower regions far away from the shower core. Over the course of an extended training, this can result in a critic that disregards these outer regions and instead focuses only on the core. This causes the decoder to place additional hits close to the border regions without penalty, resulting in artificial structures around the edges. The left panel of Figure 7.3 shows these artifacts in the outer regions.

While these artifacts are hard to notice by eye, they can have a potentially large impact on downstream reconstruction or classifier networks. Therefore we introduce an additional critic network to the BIB-AE. This critic has a structure and training approach identical to the main critic, save for the fact that it is reset after every epoch. This reset makes it so the additional critic is not blind to the outer regions due to its training history, and can easily pick up and correct any artifacts. Figure 7.3 shows the reset approach on the right panel. A comparison to the standard BIB-AE shows reduced artifacts in the outer regions.

### Modified Post Processing

The pion BIB-AE requires a Post Processor network to allow for the correct modeling of the lower part of the cell energy spectrum. This Post Processor is similar in construction to what was used in the photon shower simulation, however, the training procedure has been refined.

Three additional loss terms were added to complement the MSE and Maximum Mean Discrepancy (MMD) loss described in Section 6.3.



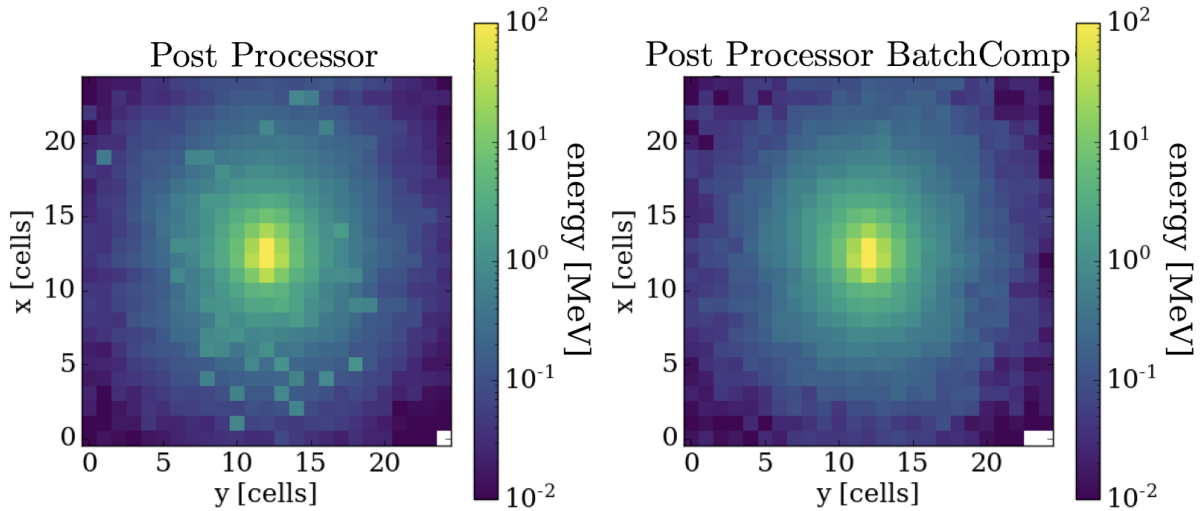


Figure 7.4: Overlays of 4000 showers belonging to 50 GeV pions, projected along the z-axis. The color scale shows the average energy in a pixel. The left panel shows the standard Post Processor and the right panel shows the Post Processor with the additional batch comparison loss. This addition greatly reduced the number of artifact pixels visible in the right panel.

The first is an additional loss term with a modified kernel resolution. This second resolution is chosen to be sensitive to a different cell energy spectrum region than the first kernel. The combination of the two MMD terms can affect a larger region of the spectrum and thereby reduces the need for fine-tuning of the kernel resolution.

The second term first flattens and sorts the Post Processor output and the real shower used as the encoder input, and then compares the two sorted vectors using an MSE and a Mean Absolute Error (MAE) loss. This has an effect similar to the MMD, as it forces the Post Processor to match the true cell energy spectrum. In addition to this, the MSE and MAE also force the post-processed shower to have the same total energy as the input, thereby further forcing the Post Processor to produce showers with the correct visible energy sum.

The third term compares the mean of one batch of post-processed showers with the mean of one batch of real showers using an MSE loss. It was found that the Post Processor has a tendency to repeatedly place hits in similar locations. This results in certain pixels around the border regions that are almost always active. This is difficult to detect in individual showers but can be clearly seen when several showers are overlaid. The loss term addresses this by comparing full batches of showers, thereby reducing the artifacts. Figure 7.4 demonstrates the difference between a Post Processor with (right) and without (left) this batch comparison loss term.

Further, the Post Processor is no longer trained in parallel to the main BIB-AE setup. It is now trained after the BIB-AE has converged, using a fixed BIB-AE model. This resulted in a more stable Post Processor training.

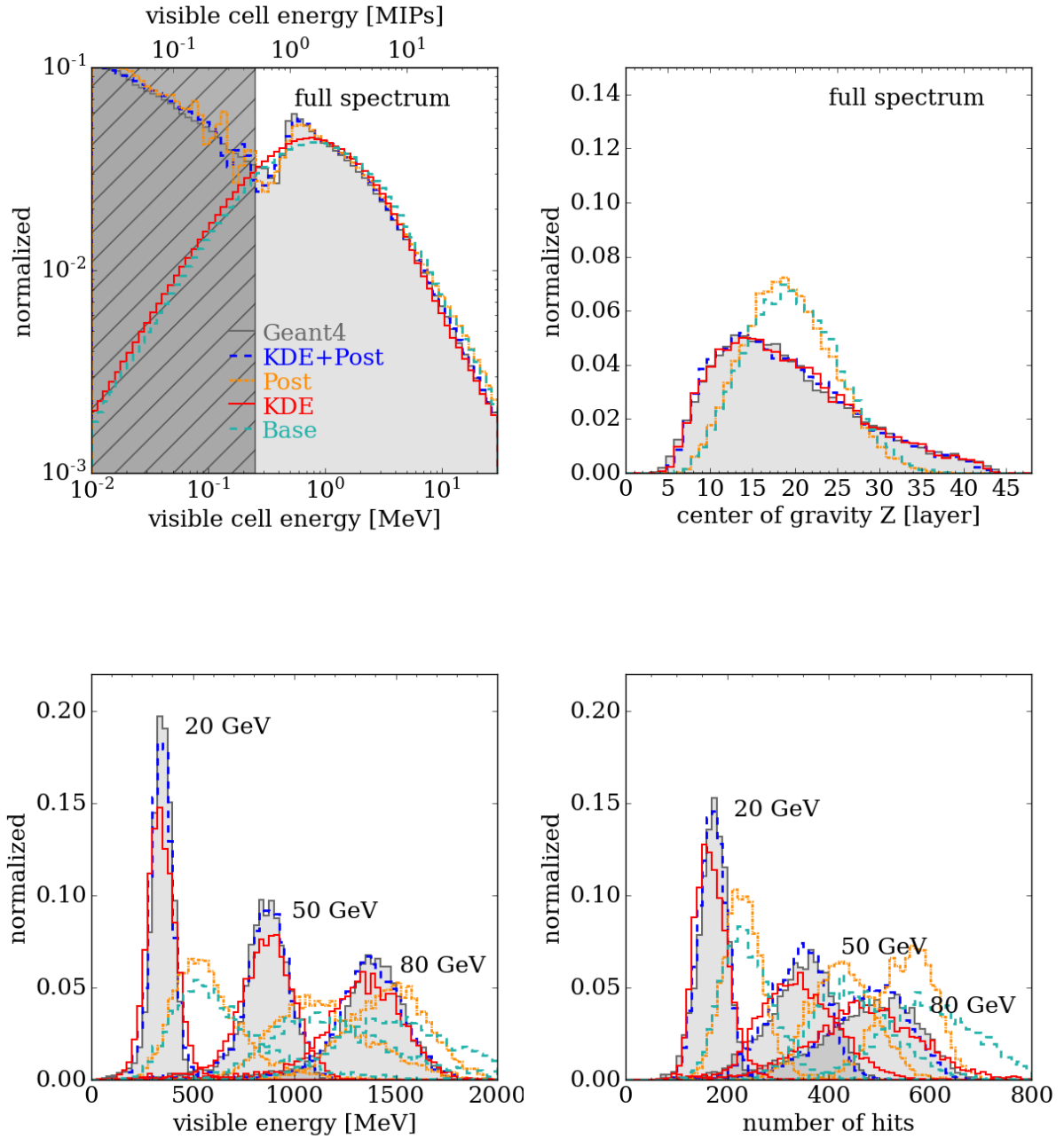


Figure 7.5: Demonstration of the effects of the KDE sampling and Post Processing for the BIB-AE. Shown lines are the default BIB-AE (turquoise, dashed), the BIB-AE with KDE sampling (red, solid), the BIB-AE with Post Processor (orange, dotted), the BIB-AE with KDE sampling and Post Processor (blue, dashed), and GEANT4 (gray, filled). The comparisons show the cell energy spectrum (top left), the center of gravity along the  $y$ -axis (top right), and the visible energy and number of hits for 20, 50, and 80 GeV (bottom right, and left, respectively). The hatched region in the cell spectrum indicates the region cut by the MIP threshold.

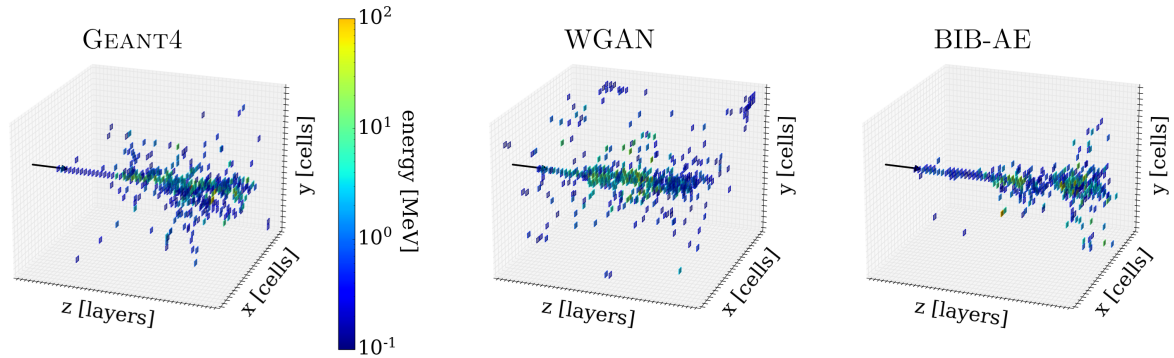


Figure 7.6: 3D renderings of individual showers that were generated using GEANT4 (left), the WGAN (center), and the BIB-AE (right). The colors of the individual pixels indicated the deposited energy in that pixel.

### Effects of the Modifications

The direct effects of the BIB-AE Post Processing and KDE sampling can be observed in Figure 7.5. The four panels show a comparison between four different versions of the BIB-AE: the default BIB-AE, labeled *Base*, in turquoise, the BIB-AE with only the added KDE sampling, labeled *KDE*, in red, the BIB-AE with only the Post Processor applied, labeled *Post*, in orange, and the BIBAE with both KDE sampling Post Processor, labeled *KDE+Post*, in blue.

The top left panel shows the cell energy distribution. Here the impact of the Post Processor is clearly visible, as both Post Processor lines accurately model the MIP peak, while the other two BIB-AE versions smooth out this feature. Further, one can see that the high-energy region is only correctly modeled by the version that makes use of the KDE sampling.

The center of gravity plot in the top right demonstrates the power of the KDE sampling, as both the baseline and post-processing-only BIB-AE fail to model this feature, resulting in a center of gravity distribution that is significantly too sharp. Meanwhile, the KDE approaches show no difficulties replicating the center of gravity with high precision.

The bottom row shows the visible energy sum and number of hits distributions for fixed energies of 20, 50, and 80 GeV pion showers. It can be seen that both non-KDE approaches once again fail to capture these distributions. However, the KDE approach without post-processing also shows noticeable deviations from the GEANT4 distributions. It is only after the Post Processor is added to the KDE sampling that the model manages to accurately describe both observables. This illustrates the importance of combining the Post Processor with the KDE sampling.

## 7.4 Results

The two modified generative models can now be compared against the test data set. As was done in the photon data set, all comparisons take place after the 0.5 MIP cutoff (0.25 MeV for the AHCAL) is applied to the showers. The comparison will be split into two parts. The first approach is directly comparable to the evaluation of the photon data and compares the outputs of the generative models directly to the outputs of GEANT4. This method, labeled *generator level* comparison, serves to give insight into how well the models can directly reproduce the data they

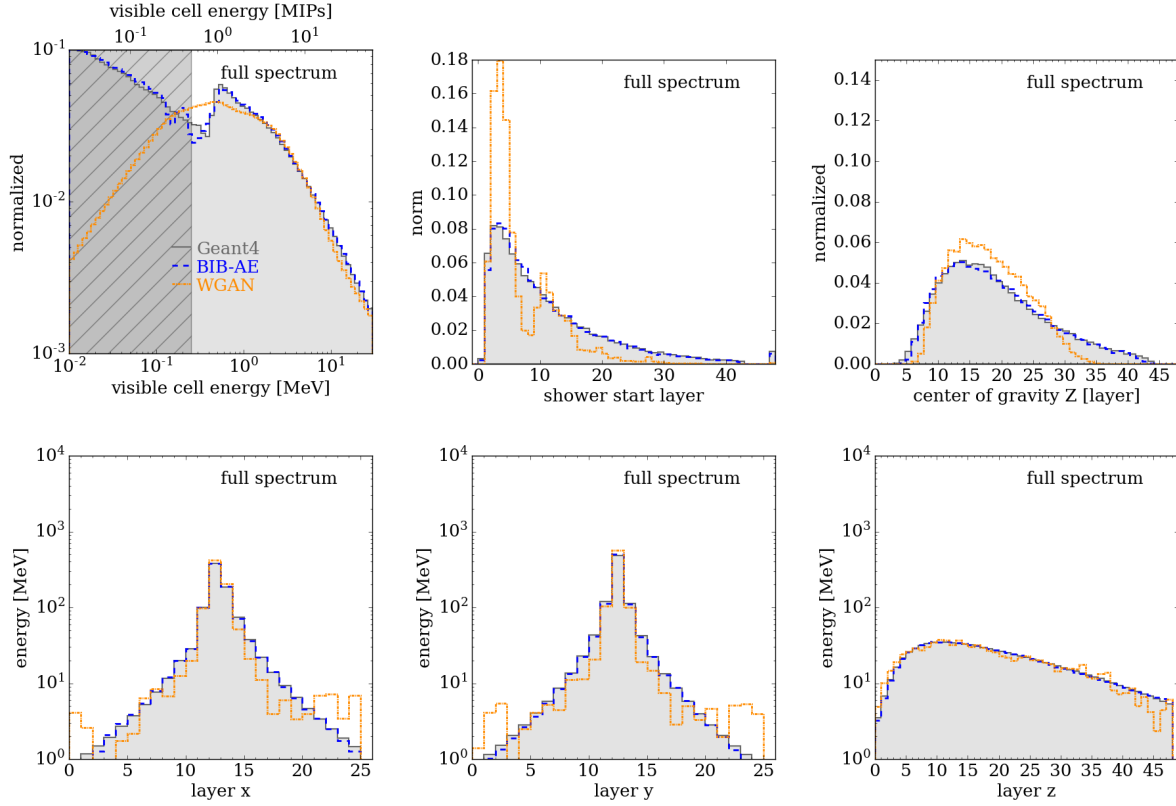


Figure 7.7: Comparison of the cell energy distribution (top, left), the shower start layer (top, center) the center of gravity in the z-direction (top, right), the transversal energy profiles in x and y direction (bottom left and center) and the longitudinal energy profile (bottom, right). The hatched region in the cell spectrum indicates the region cut by the MIP threshold. The lines correspond to GEANT4 (filled, gray), the WGAN (orange, dotted), and the BIB-AE (blue, dashed). Figure taken from [3].

were trained on. The second approach is modeled after a more realistic physics use case and first runs both the generated and test showers through a reconstruction setup. This *reconstruction level* comparison is especially important, as matching the reconstructed properties of GEANT4 is a key ingredient for bringing generative models from proof of concept to practical application.

## Generator Level

In the initial comparison, we directly inspect the shower images produced by the three approaches. Figure 7.6 shows example showers produced by GEANT4 in the left, by the WGAN in the center, and by the BIB-AE on the right panel, respectively. The BIB-AE shower shows no clear deviations from the GEANT4 example. However, the WGAN shower shows unphysical hits around the edge regions. While this is not necessarily a disqualifying point, it does indicate that the WGAN is less successful in capturing the shower structure than the BIB-AE.

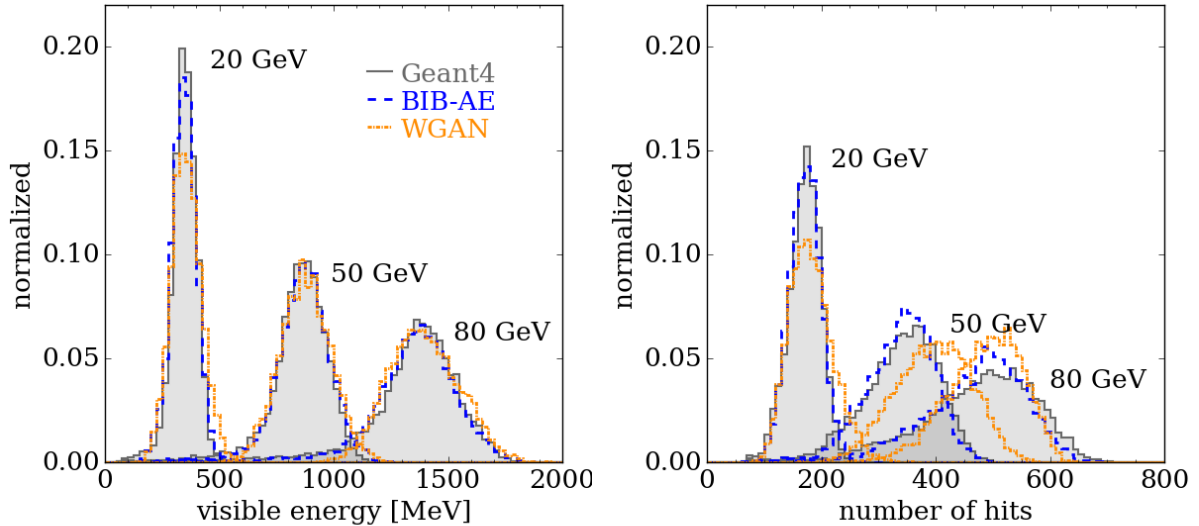


Figure 7.8: Comparisons of the visible energy sum (left) and the number of hits (right) for showers from pions with 20, 50, and 80 GeV. The lines correspond to GEANT4 (filled, gray), the WGAN (orange, dotted), and the BIB-AE (blue, dashed). Figure taken from [3].

Similar observations can be made for the marginal distributions. Figure 7.7 compares per shower variables between GEANT4 in the filled gray histogram, for the WGAN in the dotted orange line, and for the BIB-AE in the blue dashed line. The cell energy spectrum is shown on the top left panel. The hatched region shows the parts of the spectrum removed by the MIP cutoff. Similar to the photon shower result, we can see that the WGAN smooths over the MIP peak around 0.5 MeV, while the Post Processor allows the BIB-AE to capture this peak. Further one can see a slight mismatch between the WGAN and GEANT4 in the high energy part, which is not present for the BIB-AE. The top center plot shows the distribution of the starting layer of the showers [160]. While the start layers of the BIB-AE showers match up nearly perfectly with GEANT4, the WGAN showers show significant deviations, as well as a strange fluctuating structure. This can likely be traced back to the un-physical hits visible in Figure 7.6, which confuse the algorithm that determines the shower starting layer. The distribution of the center of gravity along the  $z$ -axis is plotted on the top left panel. The BIB-AE again produces showers that match the test data. The WGAN exhibits a distribution that is too sharp, somewhat similar to the result obtained by the BIB-AE without KDE sampling, as can be seen in Figure 7.5. The bottom three plots show the average energy profiles of the showers along the  $x$ ,  $y$ , and  $z$  directions. Within these profiles, one can directly see the un-physical hits around the edges in the showers produced by the WGAN. This is in line with what can be seen in the WGAN shower image in Figure 7.6. The BIB-AE reproduces all three profiles with high accuracy, even managing to emulate the asymmetry in the  $x$ -direction profile caused by charged tracks being bent by the magnetic field.

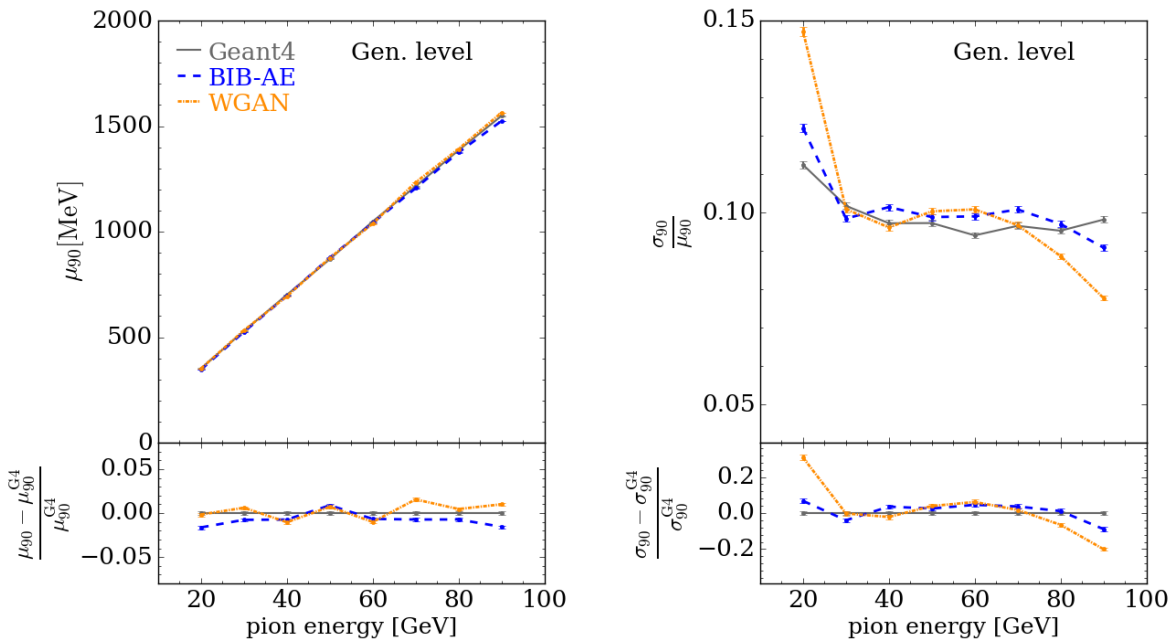


Figure 7.9: Comparison of  $\mu_{90}$  (left) and  $\frac{\sigma_{90}}{\mu_{90}}$  (right) of the visible energy sum distributions for single pion energies ranging from 20 to 90 GeV. Lower plot sections show the ratios between GEANT4 and the generative models. The lines correspond to GEANT4 (filled, gray), the WGAN (orange, dotted), and the BIB-AE (blue, dashed). Figure taken from [3].

In order to evaluate the energy conditioning, Figure 7.8 shows the visible energy sum distributions (left) and the number of hits distributions (right) for showers produced by pions with fixed energies of either 20, 50, or 80 GeV. It can be seen that both the WGAN and BIB-AE describe the energy distributions well, with the BIB-AE capturing the widths slightly better. The number of hits distributions of both generative models show deviations from GEANT4, however, the WGAN ends up with larger differences due to the artificial hits.

Similar to the photon showers in Chapter 6, we further test the conditioning by calculating the mean  $\mu_{90}$  and relative width<sup>3</sup>  $\frac{\sigma_{90}}{\mu_{90}}$  for energy distributions of 20 GeV–90 GeV pion showers. The results are shown in Figure 7.9. The  $\mu_{90}$  on the left panel is reproduced accurately by both models, with deviations in the order of 2%, an even closer match than was achieved for the photon showers. The ratio of width over mean, shown in the right-side plot, is less accurately modeled. However, the BIB-AE still manages to follow the overall trend decently well and shows deviations of only around 5% except for the 90 GeV showers. The WGAN similarly manages to accurately model the central energy values, however, it displays deviations of up to 30% in the edge regions.

The final generator level comparison examines the correlations between the shower variables. To this end, we once again calculate the pairwise Pearson correlation coefficients between the

<sup>3</sup>Note that while this quantity has similar properties to the resolution of the calorimeter, its numerical values do not represent the resolution

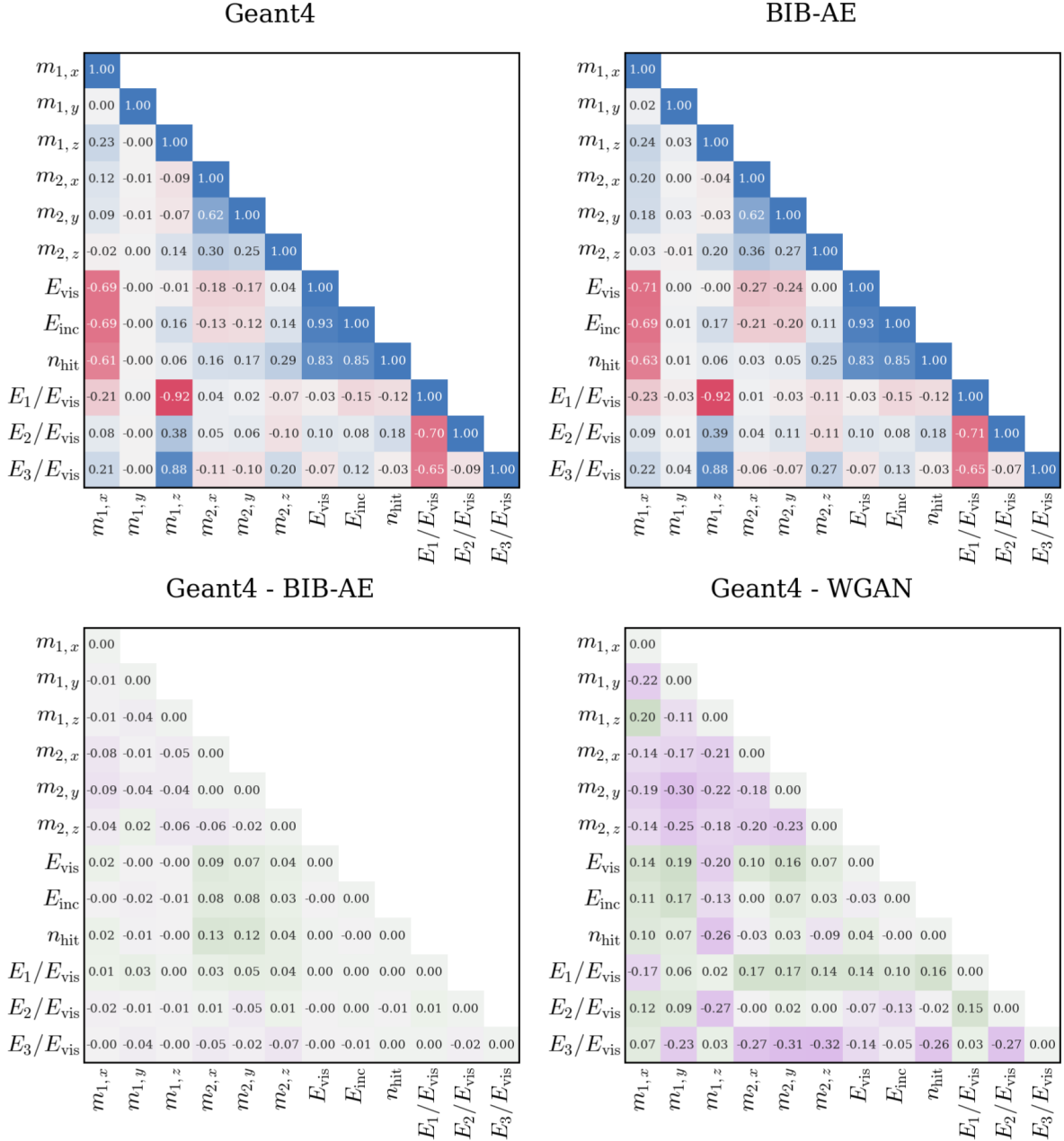


Figure 7.10: Pairwise Pearson correlation coefficients between various observables that are outlined in the text. For GEANT4 (top, left) and the BIB-AE (top, right) the full correlations are shown, for the generative models we show the difference between the generative and GEANT4 correlations. Difference matrices use identical color scales. Figure taken from [3].

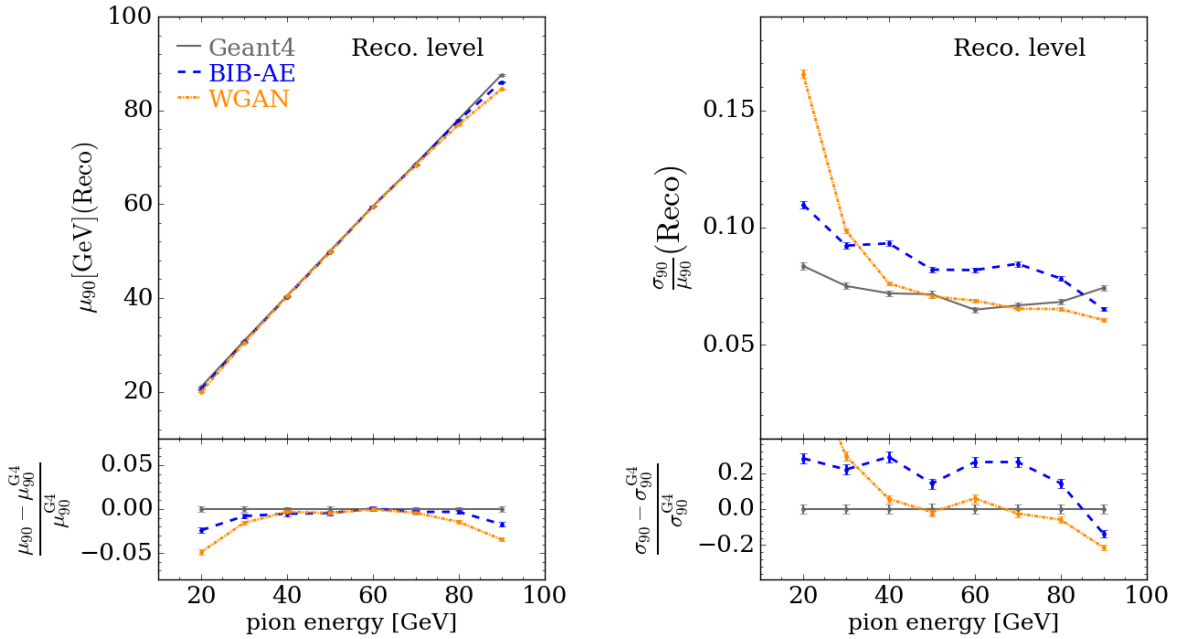


Figure 7.11: Comparison the reconstructed of  $\mu_{90}$  (left) and  $\frac{\sigma_{90}}{\mu_{90}}$  (right) for single pion energies ranging from 20 to 90 GeV. Lower plot sections show the ratios between GEANT4 and the generative models. The lines correspond to GEANT4 (filled, gray), the WGAN (orange, dotted), and the BIB-AE (blue, dashed). Figure taken from [3].

variables outlined in Section 6.4. The resulting correlation matrices of GEANT4 and the BIB-AE are shown in the top row of Figure 7.10. The bottom row compares the generative correlations to GEANT4 by calculating the element-wise difference between the matrices. For the BIB-AE (bottom, left) the majority of correlations agree with GEANT4 within  $\pm 0.05$  and the highest difference is 0.13. Compared to the photon shower results, this represents a significant improvement in the ability of the BIB-AE to learn correlations. This improvement is in large part attributable to the improved KDE sampling. The difference matrix of the WGAN (bottom right) displays significant deviations compared to GEANT4, reaching up to  $-0.32$ . This is again indicative of the difficulty the WGAN has with learning shower shapes.

## Reconstruction Level

In order to approximate the conditions of realistic physics analyses, we run a dedicated reconstruction software on the generative showers<sup>4</sup>. The ILD calorimeters are optimized for the particle flow [80] principle, meaning that one attempts to reconstruct every particle in a collision event using the sub-detectors best suited for a given task.

To run a particle flow reconstruction on the shower data it needs to be processed first. To this end, the hits obtained from either GEANT4 or the generative models are first run through a

<sup>4</sup>The reconstruction results were provided by Engin Eren.



Table 7.1: Comparison of the per shower simulation time between GEANT4, the WGAN, and the BIB-AE. All models were evaluated on a single core of an Intel<sup>®</sup> Xeon<sup>®</sup> CPU E5-2640 v4 (CPU). Additionally, the generative models were evaluated on an NVIDIA<sup>®</sup> A100 with 40 GB of memory (GPU). The generative models were evaluated using the batch sizes that resulted in the fastest evaluation. The uncertainties correspond to the standard deviation of the timings. Table taken from [3].

Hardware	Simulator	Time / Shower [ms]	Speed-up
CPU	GEANT4	2684 ± 125	×1
	WGAN	47.923 ± 0.089	×56
	BIB-AE	350.82 ± 0.57	×8
GPU	WGAN	0.264 ± 0.002	×10167
	BIB-AE	2.051 ± 0.005	×1309

digitization step [59] which accounts for hardware effects such as readout noise or varying pixel detector sensitivities. The digitized showers are then normalized to have hit energies in the unit of 1 MIP, which is then translated into an energy value in GeV.

The calibrated showers are then passed to PANDORAPFA [80], the main particle flow algorithm used by the ILD collaboration. PANDORAPFA uses state-of-the-art pattern recognition algorithms to cluster the hits and reconstruct the clusters into so-called Particle Flow Objects (PFOs). These PFOs describe the properties of the reconstructed particle that cause the shower.

In order to reconstruct charged pions, PANDORAPFA would normally expect tracker information in addition to the calorimeter data, because the particle flow approach is designed to make use of the more accurate tracking measurements. However, this is not an option for the generated showers, as the generative models only simulate the calorimeter hits. Therefore we reconstruct the pion showers as neutral hadrons, for which no tracking information is expected.

We perform this full reconstruction chain on both GEANT4 and the reconstructed data and compare the reconstructed shower energies in a way similar to the generator level visible energy sum comparison. Figure 7.11 shows the resulting  $\mu_{90}$  and  $\frac{\sigma_{90}}{\mu_{90}}$  results after reconstruction. On the left plot, one can see that the means of the energy distributions are reasonably well modeled by the BIB-AE model, displaying precise agreement for the central energies and deviating only by around 3% around the high and low energy regions. Likewise, the WGAN accurately describes the center, however, it shows larger deviations of approximately 5% close to the edges. The relative width on the right panel is not well described by the BIB-AE, exhibiting a large, systematic shift to higher widths. The WGAN again struggles to model the high and low energies, however, in the central energy region it shows impressive agreement with the relative widths of GEANT4.

This presents an interesting result considering the relatively poor performance of the WGAN at the generator level. However, a large fraction of the deviations between the WGAN and GEANT4 are caused by the extraneous outer hits, which are removed by the clustering algorithms

during reconstruction. Therefore it is reasonable that the reconstruction improves the WGAN results.

### Computational Timings

We are again interested in quantifying the speedup factor that can be achieved by the WGAN and the BIB-AE. To this end, we compare the average simulation time for one shower<sup>5</sup> in Table 7.1. Both models offer a significant speedup compared to GEANT4 when evaluated on a CPU. When the generative showers are instead sampled using a GPU, the speedup factor increases by a further  $\sim 170\times$ . For both hardware types, the WGAN is around  $7\times$  faster than the BIB-AE.

## 7.5 Conclusion

In this section, we demonstrated how we can successfully extend the generative models used during photon shower generation to the significantly more challenging task of hadron shower simulation.

We were able to make significant improvements to the generative models, foremost in the shape of KDE-sampled BIB-AE. Using this setup we were able to replicate GEANT4 shower to a high level of agreement, often showing only percent-level deviations between the distributions of GEANT4 and those of the BIB-AE. The WGAN model exhibits larger deviations than the BIB-AE, when compared to GEANT4, and has visible difficulties modeling the longitudinal and transversal profiles of the showers. However the WGAN also provides a significantly greater speedup factor compared to the BIB-AE. This presents an interesting trade-off between shower accuracy and generation speed, with the WGAN offering the better speedup and the BIB-AE achieving greater accuracy.

Additional studies were performed, taking into account the effects of a dedicated reconstruction algorithm in the form of PANDORAPFA. We find that the distributions of particle energies reconstructed from the generative showers and the GEANT4 showers show noticeable deviations, especially the relative widths of the distributions are overestimated by the BIB-AE. However, the overall means of the reconstructed energy distributions are described well by both generative models. Notably, the WGAN reaches this respectable reconstruction level performance despite its inability to accurately reproduce several generator level observables. This indicates that even imperfect fast simulators may still be sufficiently accurate post reconstruction.

---

<sup>5</sup>Timings for the generative models were provided by Lennart Rustige, timings for GEANT4 was provided by Engin Eren.

## Chapter 8

# CALICE Testbeam Data Generation

Generative FastSim approaches almost universally train a generative model on data that was simulated using classical MC methods. The resulting models can produce samples significantly faster and, as was shown in Chapter 6 and Chapter 7, produce samples that very closely match those produced by classical methods. However, while it was demonstrated in Chapter 5 that it is possible to build a generative model that describes the data distribution more accurately than the training data alone, actually achieving this on a high-dimensional, complex data set is a significant challenge.

Small deviations between truth and simulation are not immediately disqualifying factors. In fact, classical MC simulation will often show slight deviations when compared to real measurement data. However, a generative FastSim model trained on MC will compound these deviations. This places generative models in a spot where they are faster than classical simulations, but are also bound to be less accurate when compared to real data. However, there is nothing that dictates a generative model has to be trained on MC data. Since a generative model can learn any underlying distribution, it can, in principle, be trained directly on data.

This offers an exciting proposition. A generative model trained directly on measurement data will have deviations from the true data, however, these deviations can be similar to or even smaller than the deviations of classical MC simulations. At the same time, the generative approach will still be significantly faster.

There are inherent problems with simulation trained to exactly reproduce measurement data. If one intends to use the simulation to compare theoretical predictions with experimental results, then a simulation that per definition matches the experimental results is pointless. However, the multiple steps involved in the full HEP MC simulation chain have varying dependencies on the underlying theoretical model. For example, if one were to introduce a new heavy resonance, it would result in significant changes to the simulation of the initial hard scattering and to the decay modeling. However, the simulation of how the final state standard model particles interact with the detector would be unaffected. Therefore, a well-understood but computationally expensive simulation step, such as shower simulation, would be an ideal candidate for a data-trained generative simulation setup. We explore the feasibility of such a setup using testbeam data recorded with the CALICE AHCAL prototype.

This Chapter begins by briefly presenting the CALICE testbeam data in Section 8.1. Section 8.2 outlines the BIB-AE model used to model the testbeam data. In Section 8.3 a comparison

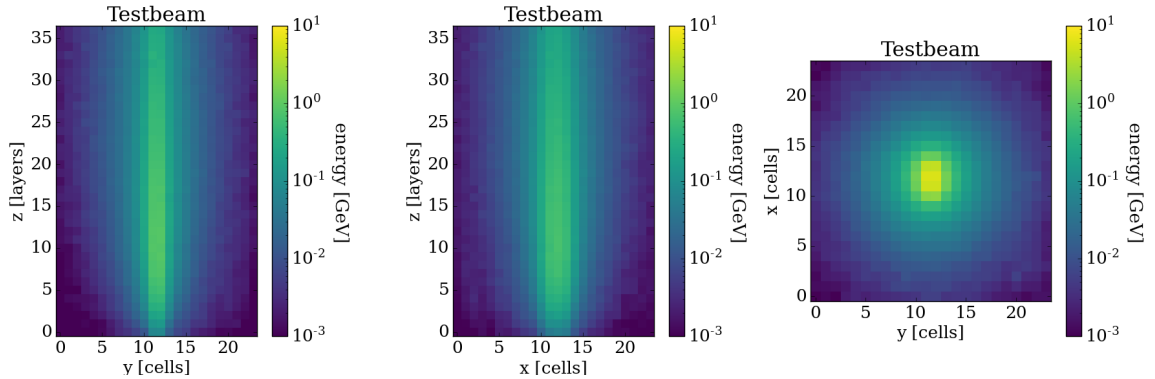


Figure 8.1: Overlays of 7500 testbeam showers produced by 80 GeV pions, projected along the x-axis (left), y-axis (center), and z-axis (right).

between the performance of the BIB-AE and GEANT4 is shown. Section 8.4 presents the obtained conclusions.

## 8.1 Testbeam Data Set

The training data used in this chapter is based on the testbeam measurements performed with the CALICE AHCAL prototype in June 2018. The tests involved measurements of charged pions with various fixed energies, provided by the Super Proton Synchrotron at CERN. Details of the testbeam runs, calibrations, and reconstructions can be found in Reference [161].

The main prototype consists of 3 parts, a pre-shower layer, the main AHCAL, and a tail catcher. We focus our efforts on the main AHCAL, which uses the same technologies as the AHCAL foreseen for the ILD described in Section 2.6. The prototype consists of 39 active layers, 38 of which are made up of  $24 \times 24$  cells. The second to last layer featured a different resolution with  $\frac{1}{4}$  the number of cells, as a test of other geometry options. To prevent this from complicating the generative training we disregard the last two layers, reducing the overall number of included layers to 37.

Measurements were performed using multiple pion energies and impact positions. For our training data, we chose runs with pion energies of 40, 60, 80, 120, and 160 GeV with impact points at the center of the calorimeter prototype. The energy depositions in the calorimeter cells were calibrated and converted into a ROOT [162] file using ILCSoft [154]. Since the individual calorimeter cells are aligned in a regular grid, we directly converted the calorimeter hits into a  $24 \times 24 \times 37$  tensor to be used for training the network. The overlays of the showers projected along the  $x$ ,  $y$ , and  $z$ -direction are shown in Figure 8.1. The energy values in the tensors have been normalized to the value of 1 MIP, equivalent to 0.0268 GeV for the prototype. Note that this differs significantly from the MIP energy value used for the photon or pion shower generation in Chapters 6 and 7. This is because the testbeam data already has a calibration factor applied that takes into account the sampling fraction of the calorimeter, which was not the case for the previously used GEANT4 data. By default the calibration and conversion script also applies a cut, removing any hit below 0.5 MIPs. However generative models often struggle to exactly reproduce

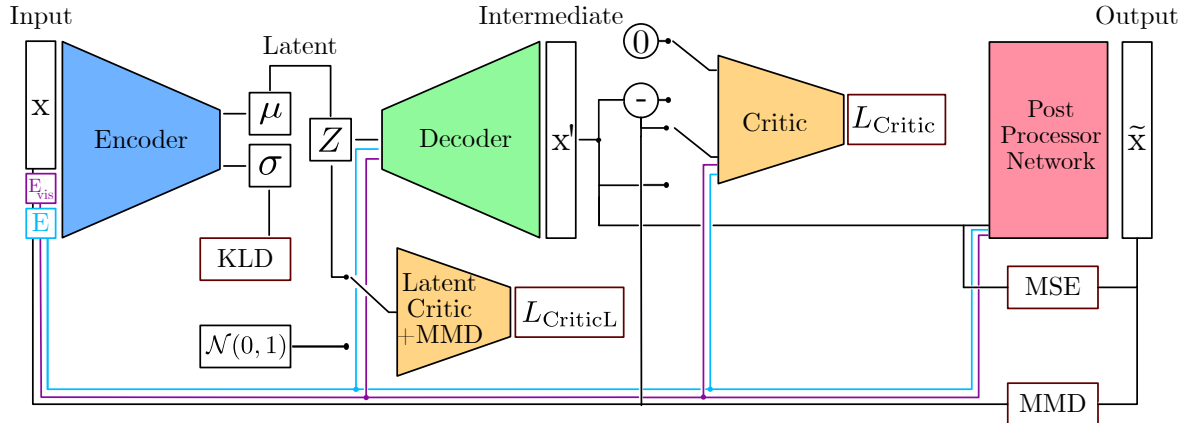


Figure 8.2: Illustration of the full testbeam shower BIB-AE setup with the visible energy sum conditioning indicated in purple. Image adapted from [2, 3].

sharp cutoffs. Therefore we lowered the threshold of this cut to 0.2 MIP for the training data<sup>1</sup>. The AHCAL prototype has a per-cell auto-trigger threshold of 0.2 MIP, and therefore effectively contains no hits below energy value. Therefore, by lowering the calibration cutoff to 0.2 MIP we ensure no existing hits are removed by the calibration. For the evaluation and comparison, we still apply the standard 0.5 MIP cut. This still leaves a sharp cutoff in the training data, however, it is in a region that is not relevant for the performance of the model. This makes it less problematic if the cutoff is mismodeled.

The full training data set consists of 50k showers of each pion energy, resulting in 250k total showers. An additional test set containing 7500 showers per pion energy was used during evaluation.

For the purpose of analyzing the testbeam measurements MC simulations were performed using GEANT4 for the exact AHCAL prototype and specific pion energies. Two sets of simulations were produced, using either the *QGSP\_BERT\_HP* or *FTFP\_BERT\_HP* physics list described in Section 2.5. Both sets were run through digitization to simulate detector effects, most notably electronic noise and the effects of the silicon photomultipliers. More details on the simulation and digitization can again be found in Reference [161].

We use these GEANT4 showers as a benchmark to compare our generative model performance. To this end, we use the same ILCSoft setup that was used for the testbeam data to convert the GEANT4 simulation into  $24 \times 24 \times 37$  tensors. The amount of GEANT4 data was chosen to be 7500 showers per pion energy, matching the test set.

## 8.2 BIB-AE Model

The BIB-AE model used to generate testbeam pion showers is adapted from the previously introduced pion generation BIB-AE. The convolutional layers of the encoder, decoder, and critics were adjusted to account for the modified input size of  $24 \times 24 \times 37$  instead of  $25 \times 25 \times 48$ . The minibatch discrimination approach is maintained, as are the dual resetting critic scheme and the

<sup>1</sup>The calibration setup was provided by Daniel Heuchel

improved Post Processor model. However, in order to accommodate the differences between the previous GEANT4 training data and the testbeam training data, as well as to try and improve the overall generation precision, an additional modification was introduced to the BIB-AE setup. A full description of the testbeam data BIB-AE architecture and its training parameters is given in Appendix A.

### Visible Energy Sum Conditioning

The CALOFLOW [163, 164] approach introduced a novel two-step approach to simulate showers in a calorimeter. It first uses a dedicated *energy flow* model to generate the total shower energy and energy per calorimeter layer for a given incident particle energy. In the second step, the main *shower flow* model uses these energies as conditional inputs to produce an appropriate shower. Finally, the produced shower is re-scaled to match the total and per layer energies produced by the energy flow.

We incorporate a similar principle into our BIB-AE setup. The main decoder, encoder, and critics are conditioned on the visible energy sum in a shower, in addition to the energy label of the true particle. This additional conditioning is indicated in purple in Figure 8.2. During generation, the decoder receives three inputs; the sampled latent space, the true energy label, and the desired visible energy sum of the shower. As these three inputs will be highly correlated, we sample them from a combined KDE. Similar to the pion generation case we again use a rejection sampling approach to generate latent space samples for specific energies. The produced showers then have their pixel values scaled such that the sum over all pixels after the MIP cutoff matches the desired visible energy sum<sup>2</sup>.

## 8.3 Results

We now compare the showers generated by the BIB-AE network to the test subset of the testbeam data set. Additionally, we also compare both the BIB-AE showers and testbeam showers to showers simulated using GEANT4. Different GEANT4 physics lists may have better or worse performance for certain features or pion energies. Therefore we use both physics lists in this comparison to ensure GEANT4 is adequately represented. However, it is not the goal of this work to compare or interpret the relative performance of the two different physics lists. The lists are labeled QGSP for the QGSP\_BERT\_HP list and FTFP for the FTFP\_BERT\_HP.

All showers used in the comparison originally have energy values in units of MIPs, however, for the comparison, these energies were translated into GeV, using the calibration factor  $1 \text{ MIP} \approx 0.0268 \text{ GeV}$ . This change does not affect any results, however, it makes the energy scale of the comparison plots more intuitive to understand. Furthermore, all showers have the 0.5 MIP cutoff applied, with a threshold of 0.0134 GeV.

As the first performance evaluation, we compare individual showers produced by all four approaches. The exemplary showers for each data set can be seen in Figure 8.3. From this figure, we can see that both GEANT4 and the BIB-AE produce showers without any obvious unphysical artifacts.

<sup>2</sup>The methods for rescaling the shower was provided by Claudius Krause.

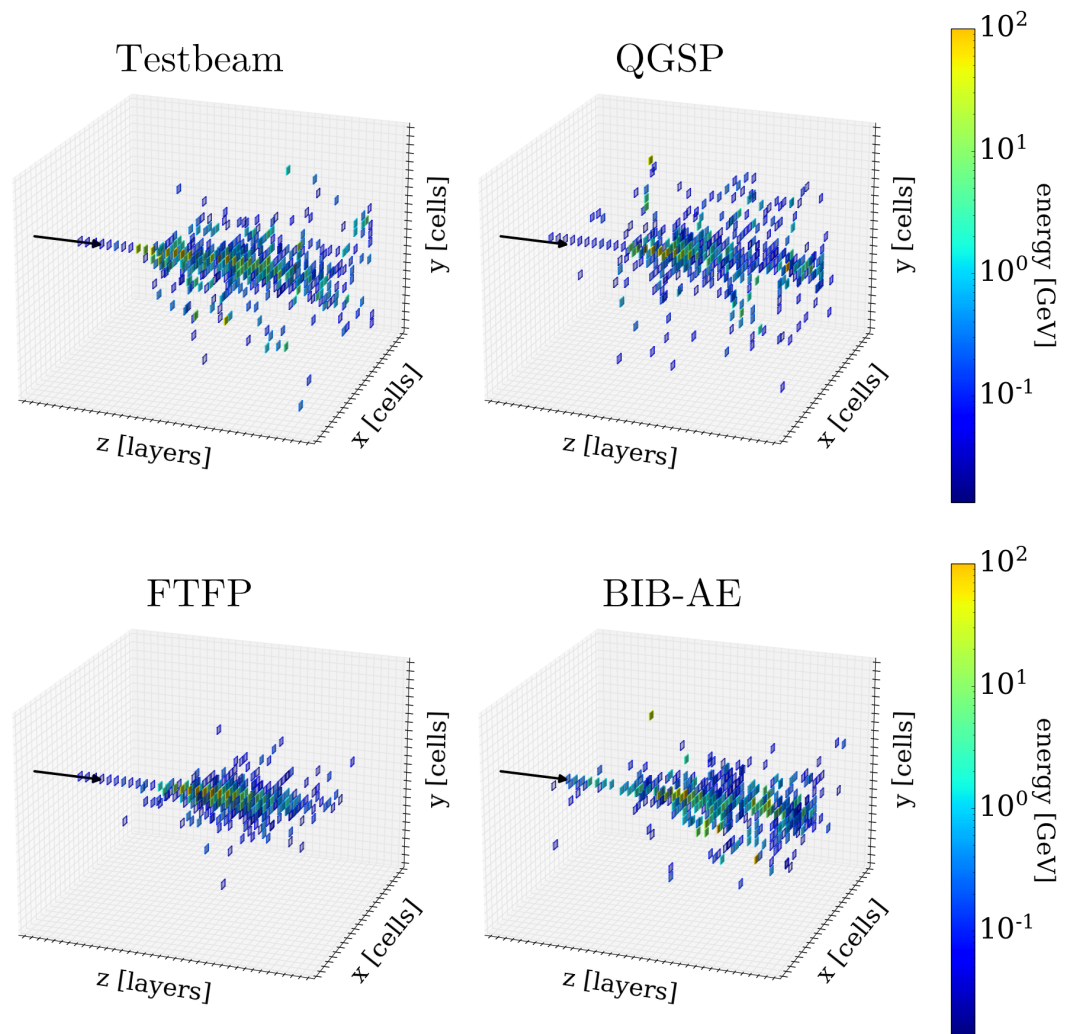


Figure 8.3: 3D renderings of individual showers that were generated using the testbeam data (top left), GEANT4 with the QGSP\_BERT\_HP physics list (top right), GEANT4 with the FTFP\_BERT\_HP physics list (bottom left), and the BIB-AE (bottom right). The colors of the individual pixels indicated the deposited energy in that pixel.

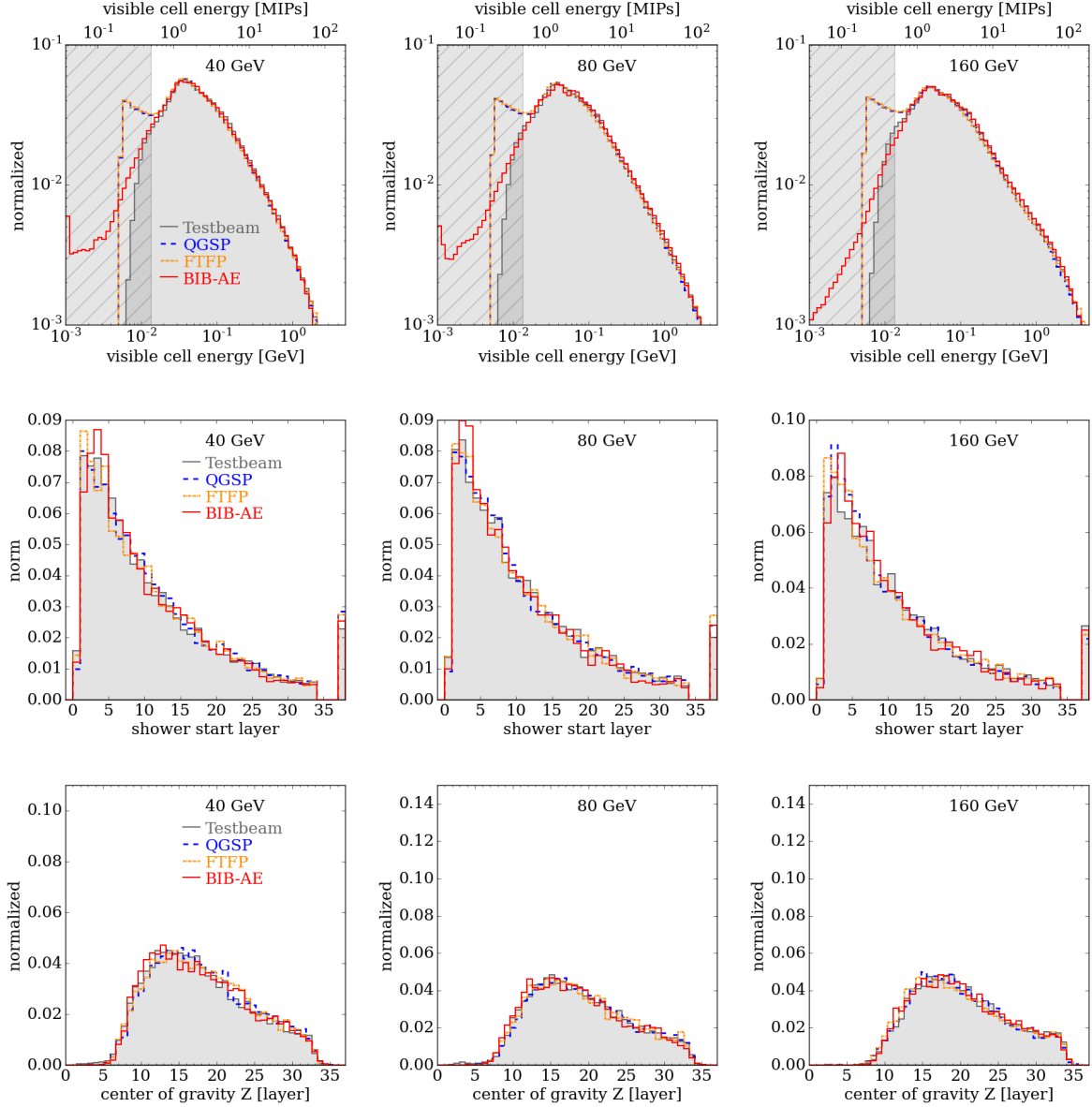


Figure 8.4: Comparisons the cell energy spectrum (top row), the shower start layer (middle row), and the center of gravity along the  $y$ -direction (bottom row). The hatched region in the cell spectrum indicates the region cut by the MIP threshold. For the shower start layer comparison, a start layer 38 indicates that no shower start layer was found. The lines correspond to the testbeam data (filled, gray), GEANT4 with the QGSP\_BERT\_HP physics list (blue, dashed), GEANT4 with the FTFP\_BERT\_HP physics list (orange, dotted), and the BIB-AE (red, solid). Comparison plots are shown for 40 GeV showers (left), 80 GeV showers (middle), and 160 GeV showers (right).



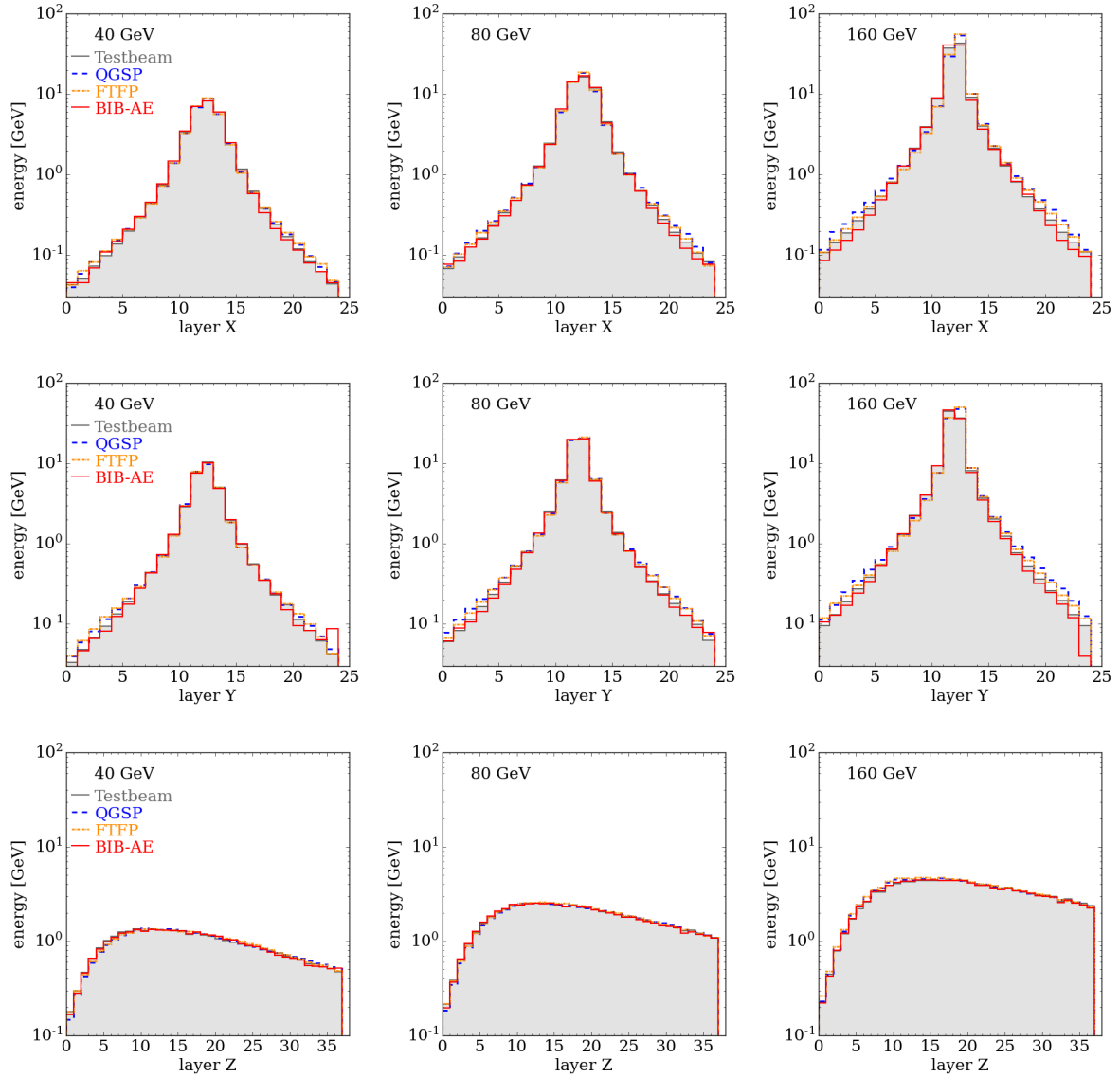


Figure 8.5: Comparisons of the shower energy profile along the  $x$ -direction (top row), the  $y$ -direction (middle row), and  $z$ -direction (bottom row). The lines correspond to the testbeam data (filled, gray), GEANT4 with the QGSP\_BERT\_HP physics list (blue, dashed), GEANT4 with the FTFP\_BERT\_HP physics list (orange, dotted), and the BIB-AE (red, solid). Comparison plots are shown for 40 GeV showers (left), 80 GeV showers (middle), and 160 GeV showers (right).

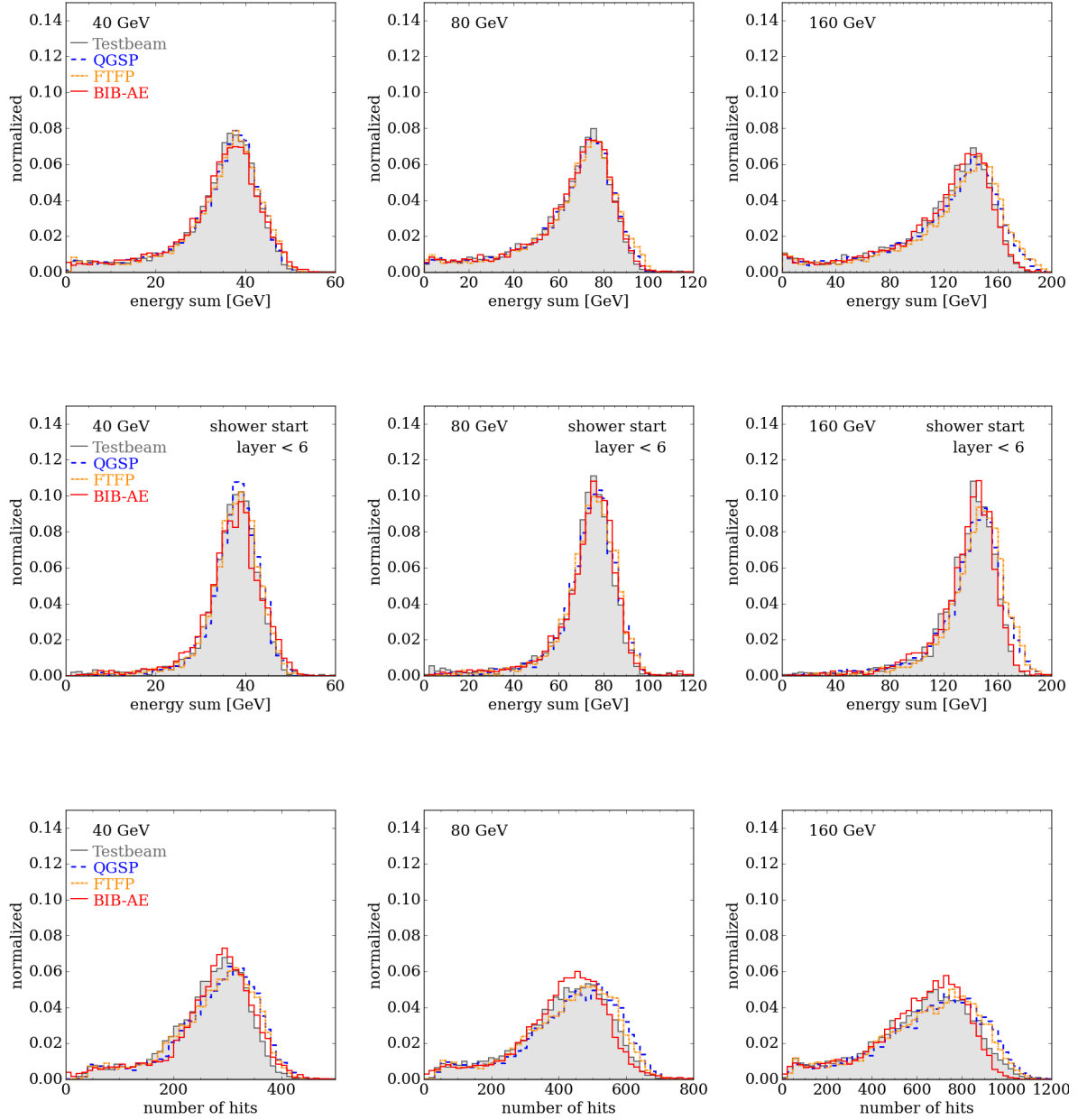


Figure 8.6: Comparisons of the visible shower energy (top row), the visible shower energy for showers that start within the first 5 calorimeter layers (middle row), and the number of hits (bottom row). The lines correspond to the testbeam data (filled, gray), GEANT4 with the QGSP\_BERT\_HP physics list (blue, dashed), GEANT4 with the FTFP\_BERT\_HP physics list (orange, dotted), and the BIB-AE (red, solid). Comparison plots are shown for 40 GeV showers (left), 80 GeV showers (middle), and 160 GeV showers (right).

## Marginal Distributions

In the next step, we compare the marginal distributions of the four sets of showers. As the testbeam data was recorded using selected beam energies, it only contains showers produced by pions with discrete energies, and a test set comprising uniformly distributed pion energies is not available. Therefore we perform separate comparisons for individual energies. Out of the five energies in the training set, we chose 40 GeV, 80 GeV, and 160 GeV, as this covers both the edge regions and central regions of the data.

The first point of comparison is the visible cell energy spectrum shown in the top row of Figure 8.4. One can see that the BIB-AE exhibits slight fluctuations and artificial structures around 0.1 GeV, especially for 80 and 160 GeV pion showers. Around the MIP threshold, it can be seen that the BIB-AE spectrum closely follows the testbeam data. The GEANT4 data shows noticeable deviations, as the per-cell trigger threshold of the AHCAL prototype is not modeled by GEANT4. Therefore, the GEANT4 spectrum starts to rise again in the low-energy region, instead of falling off. In the high-energy parts of the spectrum, all simulations achieve a good description of the testbeam data.

In the calculated shower start layer, shown in the middle row of Figure 8.4, GEANT4 achieves an accurate description for 40 and 80 GeV, while the BIB-AE has more noticeable deviations, notably around the first layers. For the 160 GeV pion example, the GEANT4 data begins to show deviations comparable to the BIB-AE data. For the bulk of the shower start distribution, all simulation approaches result in good descriptions.

The bottom row of Figure 8.4 shows the center of gravity along the  $z$ -direction. This distribution is well modeled by both the BIB-AE and GEANT4 and neither show significant deviations. Notably, all models capture the subtle secondary peak around layer 33 in the 160 GeV pion shower center of gravity distribution.

In Figure 8.5 the shower energy profiles along the  $x$ ,  $y$ , and  $z$  directions are shown in the top, middle and bottom rows respectively. For the  $x$  and  $y$ -profiles, both the BIB-AE model and GEANT4 show small deviations from the testbeam data around the outer edges. These deviations are similar in magnitude for all approaches. For 160 GeV GEANT4 shows a notable asymmetry in the  $x$  and  $y$ -profiles that is not present in the testbeam data. The BIB-AE on the other hand models this feature significantly more precisely. The profile along the  $z$ -direction is well modeled by both methods, showing no clear mismatches.

Figure 8.6 shows evaluations of the energy conditioning. In the top row, the visible energy sum is shown. Note that since the AHCAL prototype is a sampling calorimeter, it is possible for a 40 GeV pion to appear to deposit more than 40 GeV. This can occur when a shower deposits an above-average fraction of its energy in the active layers, which results in a visible energy higher than the original particle energy after the calibration is applied. From the distributions, it can be seen that for 40 GeV pion showers the BIB-AE description of the energy sum is considerably worse than that of GEANT4. For the 80 GeV pion case, both simulation methods show near-perfect agreement, with the exception of noticeable deviations present in the high energy end of the GEANT4 distribution. which are not present in the BIB-AE distribution. For 160 GeV pion showers, the GEANT4 visible energy distribution exhibits an overall shift towards higher energies. In contrast to this, the BIB-AE model maintains an accurate description even for high pion energies. The precise description of the energy sum shown by the BIB-AE can largely be attributed to the energy sum conditioning and re-scaling.

A significant part of the low energetic tail in the visible energy sum distribution is caused by showers that are not fully contained by the calorimeter. This effect, known as leakage, results in only a fraction of the energy of the particle being deposited within the calorimeter, leading to a lower visible energy sum. In order to rectify this effect, it is common practice in calorimeter development to focus on the visible energy distribution of showers that start within the first few layers of the calorimeter, more specifically in the first 5 layers of the AHCAL prototype. The center row of Figure 8.6 shows the visible energy with this shower start condition applied. It can be seen that the relative performance of GEANT4 and the BIB-AE remains largely consistent under the shower start cut, with the BIB-AE struggling with low pion energies and GEANT4 overestimating the energy of highly energetic pions.

The bottom row of Figure 8.6 compares how well the simulations describe the number of hits in the showers. It can be seen that both GEANT4 and the BIB-AE display different deviations. The number of hits simulated by GEANT4 is systematically shifted toward larger values, while the BIB-AE tends to produce too many showers with numbers of hits in the bulk of the distribution, resulting in a mismodeled distribution width.

To further quantify the energy conditioning we again calculate the mean  $\mu_{90}$  and relative width<sup>3</sup>  $\frac{\sigma_{90}}{\mu_{90}}$  of the visible energy distributions using the definitions introduced in Chapter 6. The results are shown in Figure 8.7. We differentiate between two cases. The first case, which is shown in the top row, calculates  $\mu_{90}$  and  $\frac{\sigma_{90}}{\mu_{90}}$  on all showers, while the second case, shown in the bottom row, considers only showers that start within the first 5 calorimeter layers. In both cases, the BIB-AE is capable of reproducing the mean with significantly higher accuracy than GEANT4, most notably at higher pion energies, where GEANT4 shows systematic deviation. For the relative width, the BIB-AE has noticeable difficulties reproducing the correct values for low pion energies, both with and without the shower start criterion. The relative widths produced by GEANT4 for low energy pions, on the other hand, are significantly closer to the testbeam data than those produced by the BIB-AE. At higher pion energies the accuracy with which the BIB-AE models the relative width improves, and its deviations become comparable to those of GEANT4. The results for both  $\mu_{90}$  and  $\frac{\sigma_{90}}{\mu_{90}}$  are congruent with the behavior of the BIB-AE and GEANT4 seen in the visible energy sum distribution shown in Figure 8.6.

## Correlations

As the final point of comparison, we again consider the correlation coefficients between shower features. The individual features included in the correlation calculation are identical to those described in Section 6.4. The only exception is that the incident particle energy was removed, as only single discrete incident energies are compared. The correlation factors are calculated for the testbeam data, both GEANT4 physics lists and the BIB-AE. The element-wise differences between testbeam and the simulation methods are used for comparison. Figure 8.8 shows the correlation and difference matrices for 80 GeV pion showers. One can see that GEANT4 results in an overall good description of the correlation coefficients, with all deviations staying below 0.2. Meanwhile, the BIB-AE exhibits fewer, but larger deviations, reaching up to a difference of 0.28. It is notable that the largest deviations displayed by the BIB-AE model are in correlation

---

<sup>3</sup>Note that while this quantity has similar properties to the resolution of the calorimeter, its numerical values do not represent the resolution

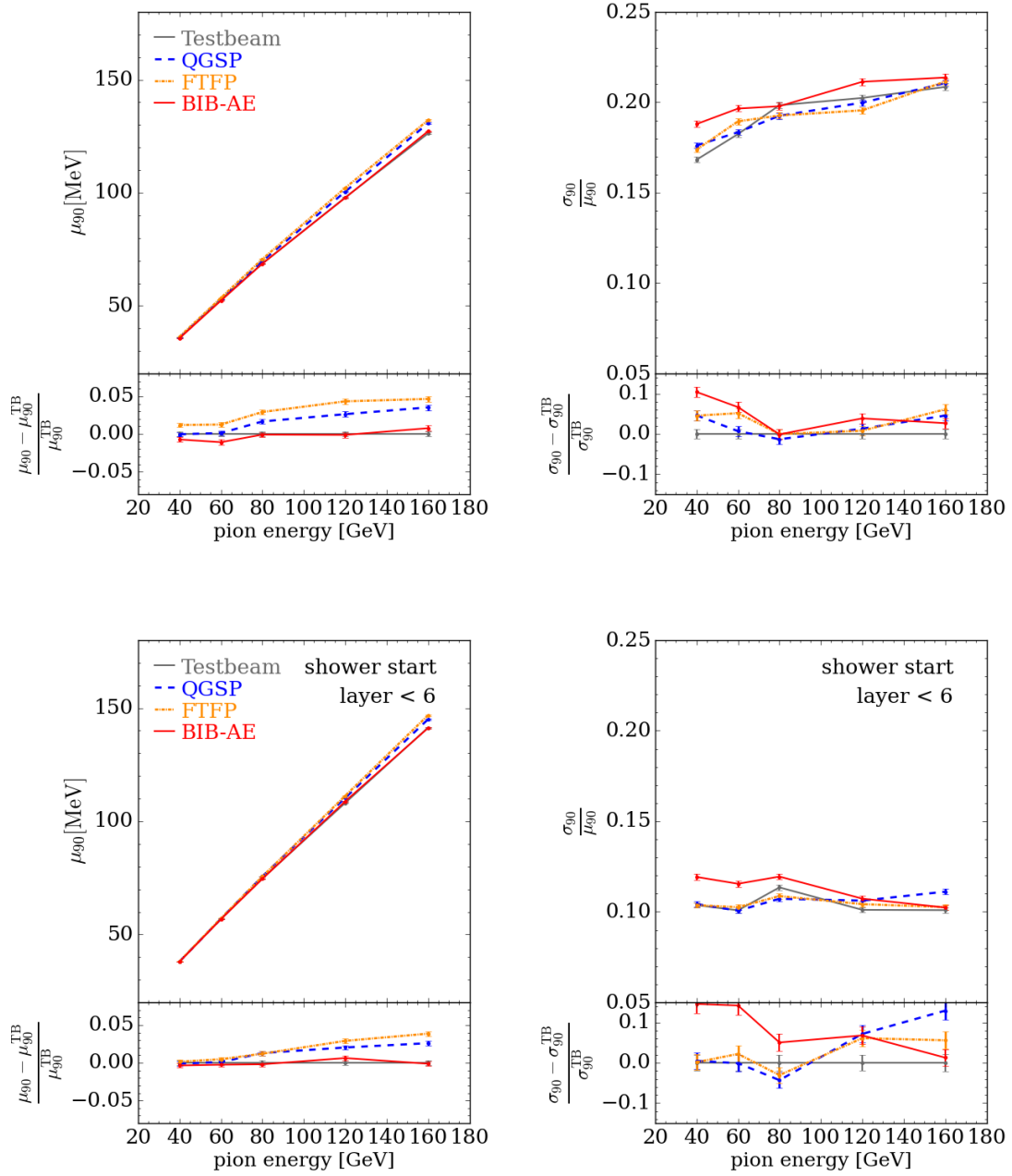


Figure 8.7: Comparison of  $\mu_{90}$  (left column) and  $\frac{\sigma_{90}}{\mu_{90}}$  (right column) of the visible energy sum distributions for single pion energies ranging from 40 to 160 GeV. The top row shows the results obtained from all showers, the bottom row shows the results considering only showers that start within the first 5 calorimeter layers. The lower plot sections show the ratios between GEANT4 and the generative models. The lines correspond to the testbeam data (filled, gray), GEANT4 with the QGSP\_BERT\_HP physics list (blue, dashed), GEANT4 with the FTFP\_BERT\_HP physics list (orange, dotted), and the BIB-AE (red, solid).

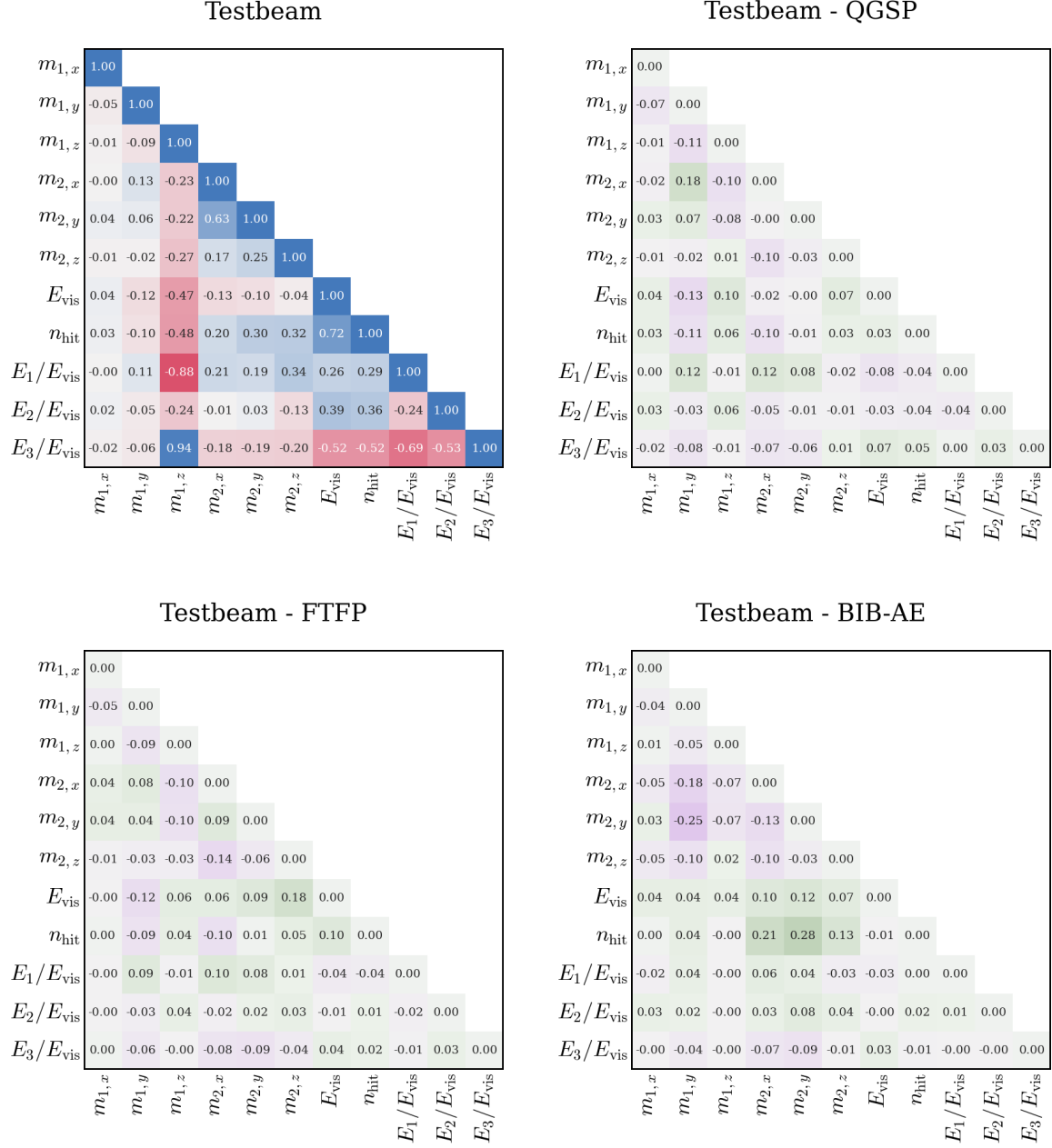


Figure 8.8: Pairwise Pearson correlation coefficients between various shower observables. For the testbeam data (top left) the full correlations are shown, for the simulation models we show the difference between the simulation and testbeam correlations. Difference matrices use identical color scales.

Table 8.1: Comparison of the per shower simulation time between GEANT4 and the BIB-AE. All models were evaluated on a single core of a Intel<sup>®</sup> Xeon<sup>®</sup> CPU E5-2640 v4 (CPU). Additionally, the BIB-AE was evaluated on an NVIDIA<sup>®</sup> A100 with 40 GB of memory (GPU). The BIB-AE was evaluated using the batch sizes that resulted in the fastest evaluation. The uncertainties correspond to the standard deviation over 10 runs. Table adapted from [2].

Hardware	Simulator	40 GeV [ms]	Speed-up	160 GeV [ms]	Speed-up
CPU	GEANT4	4475 ± 178	×1	14 544 ± 479	×1
	BIB-AE	254.41 ± 0.08	×18	253.82 ± 0.02	×57
GPU	BIB-AE	2.842 ± 0.003	×1575	2.732 ± 0.004	×5324

coefficients that involve the number of hits per shower. Therefore it can be assumed that these deviations relate to the mismodeling of the number of hits distribution shown in Figure 8.6.

### Computational Timings

In order to compare the simulation times, we rerun a small timing study using the existing GEANT4 simulation setup<sup>4</sup> within a controlled environment and evaluate the BIB-AE within the same environment. Table 8.1 shows the resulting average times per shower for GEANT4 and the BIB-AE. We differentiate between timings for the low-end (40 GeV) and high-end (160 GeV) of the pion energies contained in the testbeam training set. Similar to what was observed in Section 6.4, GEANT4 exhibits a strong dependency between simulation times and pion energies, as showers caused by particles with more energy require more interactions to be modeled. The BIB-AE has no such dependency and achieves a speedup factor ranging from ×18 for 40 GeV pions, to ×57 for 160 GeV, when evaluated on a CPU. If the BIB-AE is instead run on a GPU, the speedup factors increase to approximately ×1500 and ×5000.

## 8.4 Conclusion

We have demonstrated the feasibility of training a generative BIB-AE model with real measurement data, using AHCAL testbeam data taken by the CALICE collaboration. The resulting generative model exhibits an agreement with the true data that is comparable to the agreement achieved by GEANT4, and in some features, even demonstrates a more precise description than the classical simulation. The BIB-AE accomplishes this while still providing a factor of ×18 to ×57 speedup compared to GEANT4 on identical hardware.

This presents an exciting new possibility. While training a generative model to perfectly match contemporary MC simulations can be exceedingly difficult, training a model on data in order to reach an accuracy comparable to that of classical MC simulations is significantly more feasible.

<sup>4</sup>The GEANT4 simulation setup and the environment was provided by the CALICE collaboration.

More work is, however, required before this can be brought into practical application. Most importantly the KDE used for sampling the latent space and shower energy does not allow for interpolation. More advanced density estimation methods such as normalizing flows may provide a solution to this problem. Additionally, the interpolation behavior of the BIB-AE model itself will need to be explored to ensure it can generate showers for energies it has not seen during training. Should this prove to be difficult, a hybrid training that uses GEANT4 data as a baseline and then refines the model using real data may be a viable option.

In conclusion, this chapter represents one of the first applications of generative models to real measurement data for the purpose of fast simulation and is an interesting first step in this direction.



## Chapter 9

# OnlineFlow

The work presented in this chapter has been previously published as Reference [4] in collaboration with Anja Butter, Gregor Kasieczka, Benjamin Nachman, Tilman Plehn, David Shih, and Ramon Winterhalder. Several figures presented as well as the text are similar or identical to the content of this article. My contribution to the publication comprised implementation and optimization of the generative flow model, development of the online training methods, performing of the bump hunt and anomaly detection comparison, writing sections of the paper, handling the peer-review processes, and addressing referee comments.

The explorations of generative applications on particle physics covered in the preceding chapters were almost exclusively focused on their use as fast simulation tools. However, the ability to accurately learn the underlying distribution of a given data set has the ability to address a wide range of HEP challenges beyond fast simulation.

One such challenge is the truly staggering rate at which modern collider experiments produce data. For example, both of the general purpose LHC detector experiments each measure around 40 terabytes worth of data every second [165, 166]. This is significantly more data than can be stored for a multitude of reasons, ranging from limited available bandwidth, over limited storage space or write speed to limited computing power. However, only a small fraction of the recorded collisions contain processes relevant to current HEP research. Therefore every LHC experiment [167–170] makes use of triggers to select these interesting events for storage and discard the rest.

These triggers generally work in a two-step approach, the first, so-called L1 trigger operates on hardware level. This allows the L1 trigger to perform an extremely fast event pre-selection, however, this also makes it impossible to run any complex reconstruction. The L1 trigger sufficiently reduces the event rate so that the second level trigger, known as the high-level trigger (HLT), can perform sophisticated software-based reconstruction and select events for final storage based on this reconstruction. If both the L1 trigger and HLT do not reduce the rate of a given event class enough for all events of that class to be stored, one applies a prescale factor. This effectively discards enough events at random so the remaining ones can be stored.

Several approaches to enhance trigger systems with ML methods have already been put into use, such as the use of classification networks on HLT level at CMS [31]. Beyond this, work is ongoing on using extremely fast field-programmable gate arrays (FPGAs) to bring these ML methods to the L1 trigger as well [171–177].

One risk that remains in the current trigger paradigm, is that the trigger setup might not cover every region relevant for the search for new physics and that new discoveries might be hidden in phase-space sections currently discarded by the triggers. One approach that aims to address this concern is data scouting [178–181], where a scaled-down reconstruction is performed at trigger level and only the significantly less storage-intensive reconstruction results are saved, rather than the full raw data.

In this chapter, we explore the use of a different strategy: instead of saving individual events, we train a generative model in an online setting to learn the underlying distribution of the events. This concept is comparable to fitting a function to the measurement data and then refining the fit as more data is measured. However, by using a generative model instead of a fixed function, we do not have to constrain ourselves to a specific underlying model, as a sufficiently complex generative model will be able to approximate any given function. The main selling point of this online trained generative model is its fixed storage requirement. In the current trigger system storing more information requires proportionally more storage space, however, the storage size of the generative model is dictated only by the number of network parameters and therefore independent from the number of events that are encoded within these weights.

In principle, a sufficiently complex and expressive generative model should be able to accurately capture the entirety of LHC measurement data and thereby allow for saving information about every measured event. This is, of course, a highly unlikely scenario, for now. Therefore, we focus on the potential application of such a generative model as a scouting tool on HLT level.

This Chapter is organized as follows. We initially discuss the concept of online ML training and present the ONLINEFLOW model in Section 9.1. In Section 9.2 we demonstrate the ONLINEFLOW model on a 1-dimensional emulated invariant mass spectrum. Section 9.3 evaluates the performance of the ONLINEFLOW on the LHCO anomaly detection benchmark data set [34]. We present our conclusions in Section 9.4.

## 9.1 Online Training

Online algorithms are a specific subset of computing algorithms that operate on a sequential input [182], meaning they have to make decisions based on a given data point without knowledge about subsequent points. For our purpose, an online trained ML mode refers to a model that does not repeatedly iterate over a specific data set but is instead optimized on each data point only once. Such an online model will require significantly more data to train, however it can be trained without needing the entire data set to be stored and readily available. This makes an online model an ideal candidate to be trained on trigger level, where data is abundantly available, but the data also cannot be easily stored.

Our proposed scouting method uses such an online model trained on trigger level to extract information from phase-space regions currently ignored by trigger setups. We initially envision a model trained on all data points that reach the HLT level, however, should this prove to be infeasible, an alternative approach would be to train the model on events that pass the HLT, but whose event classes have such high rates that they require prescale factor.

In either of those two cases, the model would be trained online for a certain time, e.g. one run. Once this is complete, the trained model is moved into an offline setting and used to generate events from the underlying distribution it has learned. These events are then analyzed for any indication of new physics. It might be possible to directly discover new physics in the generated

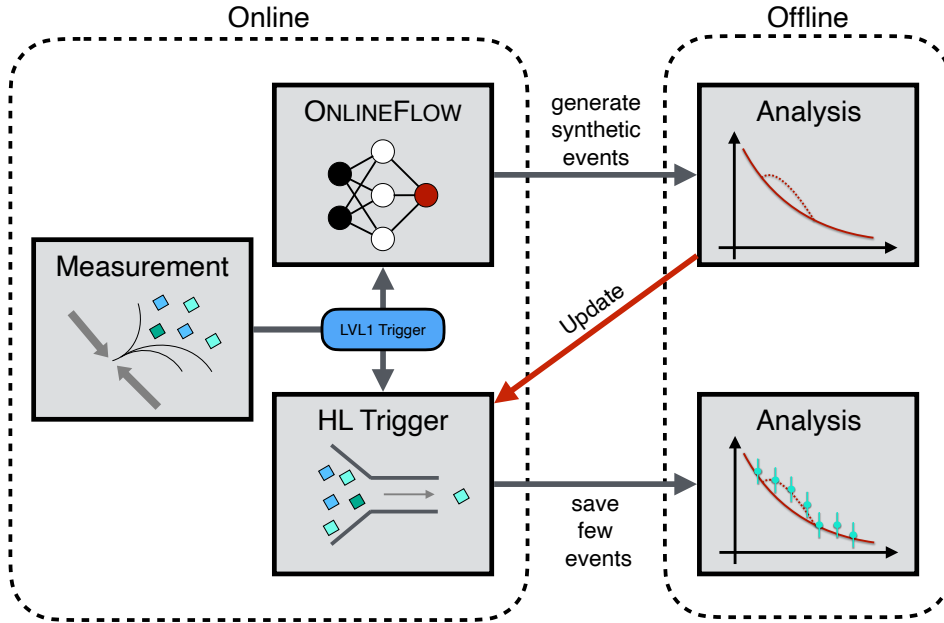


Figure 9.1: Proposed workflow of the online generative model. The model is initially trained online in parallel to the current trigger setup and then analyzed offline for any hints of new physics, which are then used to update the next run trigger menu. Figure taken from [4].

data, however, we propose a more conservative approach where any indication of an anomaly is instead used to adjust the trigger menu of the next run to specifically look for this anomaly. This concept is further illustrated in Figure 9.1.

From a ML point of view, the constraints of online training present a challenge. Pure online training would involve updating the model using individual points as they come in. This makes it difficult to utilize GPU parallelization for fast training. We, therefore, propose a hybrid approach outlined in Figure 9.2. Under this approach, the incoming data is stored in a temporary buffer with size  $N_{\text{buffer}}$ . Once this buffer is filled, it is passed to the network which begins to iterate over the buffer  $N_{\text{iter}}$  times, using batches with size  $N_{\text{batch}}$ . While the model is using the first buffer to train, a second buffer is filled with data. Once the network finishes iterating over the first buffer, the first buffer is discarded and the second buffer takes its place. This approach has two advantages, for one it allows the full utilization of GPU acceleration, and in addition to this, it allows for the synchronization between the incoming data and the model training through adaptive adjustment of  $N_{\text{iter}}$ . This ensures the model has no downtimes during low data rate segments, as it can be set to perform more iterations per buffer.

Another difficulty of online training is that the model tends to be biased toward the data point it was most recently optimized on, thereby essentially forgetting past information. To prevent this behavior we employ stochastic weight averaging [183]. The method keeps a running average over all network weights during training and these averaged weights make up the final model parameters, thereby ensuring the information from early data points is preserved in the

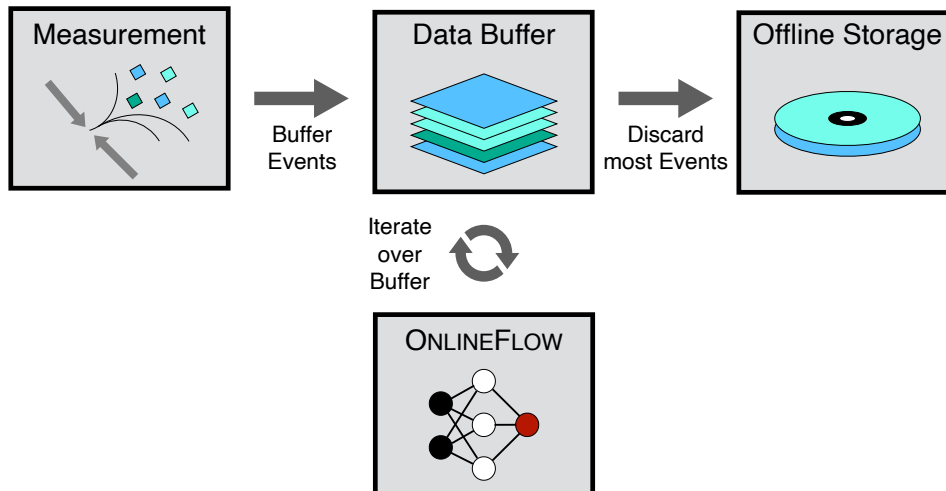


Figure 9.2: Depiction of the online training procedure. Incoming measurement data is put into a buffer. Once the buffer is full, the online network is trained using this buffer. This continues until the next buffer is filled and the initial buffer is replaced with the new buffer. Figure taken from [4].

final model. The running average is updated every time the model finishes iterating over a buffer. Not included in the averaging process are the first initial batches, as the network has not yet sufficiently converged at that point.

Finally, the online setting also dictates the type of generative model that can be used. The long intended training times are problematic for any model prone to diverge or stall due to instability in the training. This makes adversarial approaches such as a GAN or a BIB-AE unattractive. Further, the model needs to capture the features of the data set with extreme precision, thereby making normalizing flows (NFs, also see Section 4.8) a promising candidate, as they have demonstrated an ability to learn and replicate physics distributions with high accuracy [163, 164, 184–192]. While the principles discussed in this chapter are largely applicable to any kind of generative model we will employ an NF-based approach and therefore call our model **ONLINEFLOW**.

## 9.2 Bumhunt Data Set

We initially explore the **ONLINEFLOW** approach using a 1-dimensional, well-understood data set. The set is modeled after an invariant mass spectrum, consisting of an exponential falling background  $p_B$  and a Gaussian signal peak  $p_S$ , defined as

$$p_B = \frac{1}{b} \exp(-bx) \quad (9.2.1)$$

$$p_S = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad (9.2.2)$$

where  $b$  defines the fall-off rate of the background,  $\mu$  is the position of the peak, and  $\sigma$  is the peak width. The data set has been split into signal and background by giving each data point

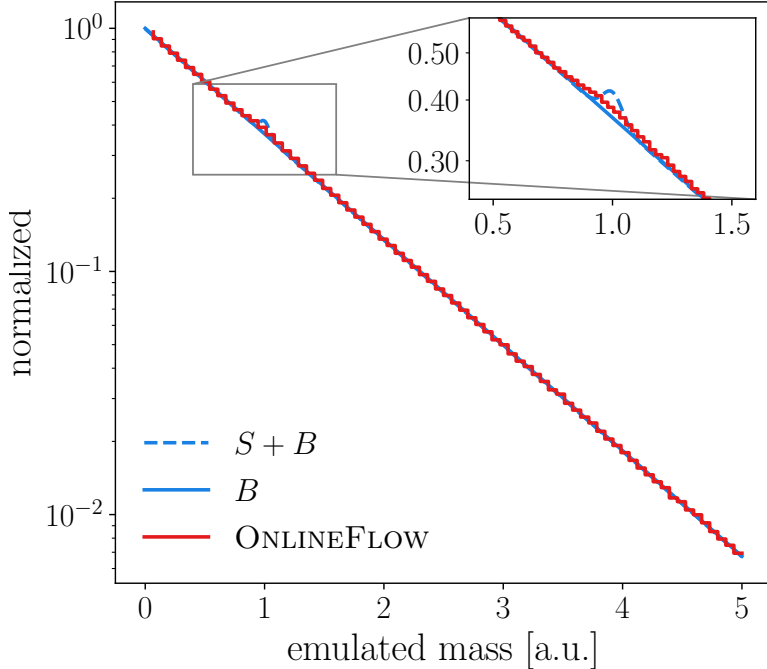


Figure 9.3: The 1-dimensional emulated invariant mass spectrum used in the initial test. Figure taken from [4].

a probability of  $\lambda$  to be drawn from the  $p_S$  and of  $1 - \lambda$  to be drawn from  $p_B$ . This was done to ensure the number of signal points in any given data set is not fixed. On average the signal fraction is equal to  $\lambda$ . For this data set the chosen parameters were  $b = 1$ ,  $\mu = 1$ ,  $\sigma = 0.04$  and  $\lambda = 0.005$ . The resulting distribution is shown by the blue line in Figure 9.3.

The ONLINEFLOW model used for this data set is a Masked Autoregressive Flow [143] (see also Section 4.8, made up of five Maskedæfor Distribution Estimation (MADE) [193] blocks. The transformation function in each block is defined by a fully connected network with two 32-node layers. MAF architectures struggle with 1-dimensional data sets, as the invertible transformations that form the basis of the setup are not clearly defined with only one input variable. Therefore the input dimensionality of the ONLINEFLOW was quadrupled from one to four, with the first dimension being the invariant mass and the remaining three consisting of Gaussian noise. This input size was found to result in stable training. The final network was implemented in PYTORCH [151] and has a total of 7560 trainable parameters.

The sharp edge at  $x = 0$  in the data set presents a further difficulty for the MAF network. Therefore we apply a logarithmic transformation to the training data to remove the edge. The ONLINEFLOW was trained using the ADAM optimizer [108] with a learning rate of  $10^{-5}$ , utilizing the hybrid online approach outlined previously. The parameters of the online training were  $N_{\text{buffer}} = 10k$ ,  $N_{\text{batch}} = 250$  and  $N_{\text{iter}} = 100$ .

The subsequent analysis of the mass spectrum requires uncertainty estimation on the data generated by the ONLINEFLOW. To this end, we use an ensemble of 20 flows with identical

architectures, trained on bootstrapped versions of the data set. This effectively encodes the uncertainty of the data set in the variance of the individual flow predictions. Typically, bootstrapping would require generating multiple re-sampled versions of the data set and training each ONLINEFLOW model with one of these re-sampled sets. This is, however, not feasible as the online training means the complete data set is never available. Therefore each data point is given a training weight sampled from a Poisson distribution with mean 1, which is independently sampled for each ONLINEFLOW in the ensemble. This has the effect of the ONLINEFLOWS seeing some points *never* (weight of 0) and some points *multiple times* (weight  $> 1$ ), emulating the bootstrapping.

As this is a proof of concept example, we store the state of the SWA after every buffer and save every data point used during training. Neither of those would be an option in a real application, however, it allows us to directly evaluate the performance of the network over the course of the training.

The result of one ONLINEFLOW training can be seen in Figure 9.3. The zoom-in on the bump position shows that the flow does not correctly reproduce the signal width, instead producing a significantly wider abundance. However, the goal of the ONLINEFLOW is not to directly discover new physics, but instead to identify potentially interesting phase-space regions to explore in detail. Therefore, the exact shape of the signal peaks is not a large concern, as long as the peak is still detectable.

## Bumphunt on Training Data

We evaluate how well the signal bump can be detected in the ONLINEFLOW data compared to the training data. Here we distinguish between the two proposed application cases. The first case trains the ONLINEFLOW on all data that passes the L1 trigger. In this situation, no data is taken in the regions that the ONLINEFLOW is trained on, therefore the ONLINEFLOW can be considered successful if it can find the signal excess with any reasonable significance. In the second case the ONLINEFLOW is applied after the HLT, but before the prescales. It is unlikely for the ONLINEFLOW to perform as well as taking data with a prescale factor of 1 (equivalent to having all data pass the prescale), however for larger factors the ONLINEFLOW may result in better sensitivity than classical data taking. Therefore we need to compare the ONLINEFLOW significance to the significance obtained for various prescale factors, so the crossover point at which the ONLINEFLOW outperforms the classical approach can be determined. Therefore, we first perform a bump hunt analysis on various fractions of the training data to establish a benchmark.

The first step consists of fitting a background function to a histogram containing  $N_{\text{pre}}$  points of the training data.  $N_{\text{pre}}$  is the number of training points after the prescale factor  $f_{\text{pre}}$  is applied, defined as  $N_{\text{pre}} = \frac{N_{\text{train}}}{f_{\text{pre}}}$ . In this case, the total number of points in the training set  $N_{\text{train}}$  is 5 million.

The background function used in the fit is a falling exponential function with several higher order terms, given by

$$p(x) = \alpha e^{-\beta x + \gamma x^2 + \delta x^3 + \epsilon x^4 + \zeta x^5}, \quad (9.2.3)$$

with fit parameters  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\delta$ ,  $\epsilon$  and  $\zeta$ . This fit function leads to a high-quality fit ( $\chi^2/\text{dof} \approx 1$ ) on both the training data and the subsequently discussed ONLINEFLOW data. We investigated

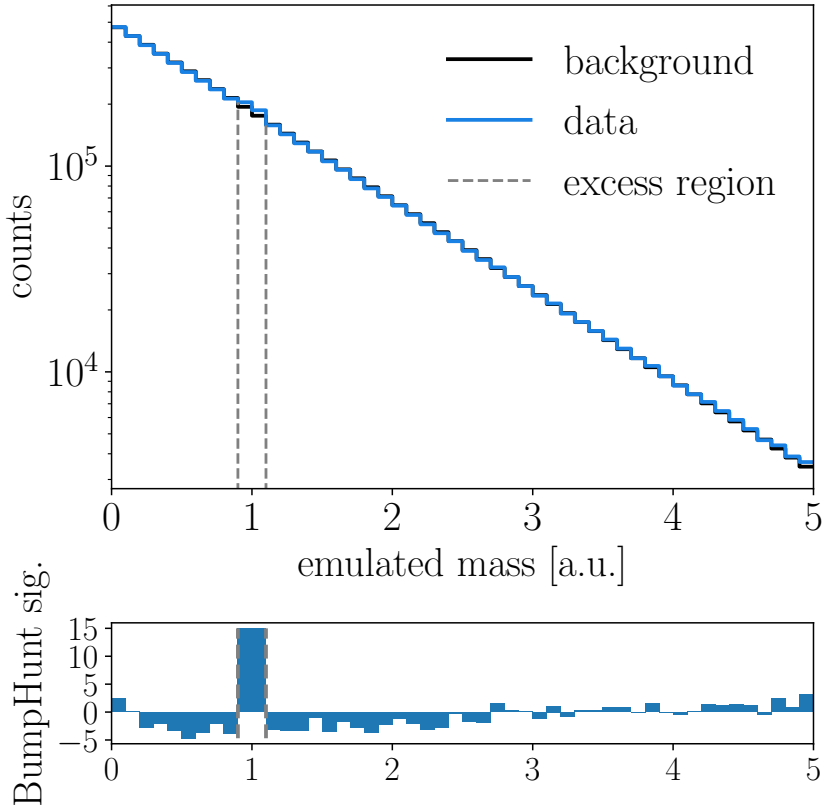


Figure 9.4: Output of the BUMPHUNTER on the 1-dimensional training data. The gray dashed lines depict the edges of the signal region. In the lower panel, the significance internally calculated by BUMPHUNTER is shown. Figure taken from [4].

a number of possible fit functions, however, our obtained results remained largely consistent regardless of the chosen function.

The resulting background estimate and data are fed into the BUMPHUNTER [194] algorithm, which scans the region between emulated mass values 0 to 5 for the largest excess between signal and background. For the purpose of this scan, the mass range is split into bins with a width of 0.1, and the minimal and maximal sizes of the signal region are set to 2 and 6 bins respectively. The BUMPHUNTER output in the data with  $f_{\text{pre}} = 1$  is shown in Figure 9.4. The region enclosed by the gray dashed lines in the signal region is in agreement with the true bump position.

The signal region determined by BUMPHUNTER is then used to determine the signal significance, defined as

$$\text{significance} = \frac{\mathcal{O} - B}{\sqrt{B}} \equiv \frac{S}{\sqrt{B}}, \quad (9.2.4)$$

where  $\mathcal{O}$  is the number of observed events in the signal region,  $B$  is the predicted background in the signal region and  $S$  is the difference between observation and prediction.

## ONLINEFLOW Significance

The ensemble of  $N_{\text{ens}} = 20$  ONLINEFLOW models is trained on the exact data used in the classical analysis using the bootstrapped online training method previously described. The significance calculation for the ONLINEFLOW is performed similarly to the bump hunt on the training data. We initially use the flows in the ensemble to each generate 10 million samples. These samples are then combined into one large set, on which the background fit using Equation (9.2.3) is performed.

This background estimation is then fed into BUMPHUNTER along with the ONLINEFLOW data to determine the signal region, which is then used to determine the significance.

As with any ML training, the ONLINEFLOW has the risk of overfitting to statistical fluctuations in the training data. This can potentially lead to structures in the generated data that could be mistaken for signal excesses. To reduce the risk of this we split the ensemble into two parts, each containing  $\frac{N_{\text{ens}}}{2} = 10$  ONLINEFLOW models. The first set of models is used in BUMPHUNTER to determine the signal region, while the second set is used to determine the signal significance in that region. This acts as cross-validation and ensures that a significant excess can only be found if it is present in more than one model.

The significance calculation for the ONLINEFLOW data is considerably more involved than for the classical bump hunt. The training data could be assumed to be Poisson distributed, allowing us to determine the background uncertainty using  $\sqrt{B}$ , however, this is not applicable to the ONLINEFLOW data. Since the ONLINEFLOW theoretically allows for an infinite number of samples to be generated, it would be trivial to generate sufficient samples to make  $\sqrt{B}$  negligible and obtain an extremely large significance.

Therefore we quantify the ONLINEFLOW significance using the variation within the second part of the ensemble. To this end we first determine the observed events  $\mathcal{O}_i$  and expected background  $B_i$  for each flow independently, where  $i$  runs from 1 to  $\frac{N_{\text{ens}}}{2}$ . These individual contributions are then combined into an average number of observed events  $\mathcal{O}$  and average expected background  $B$ , defined as

$$\mathcal{O} = \frac{2}{N_{\text{ens}}} \sum_i^{\frac{N_{\text{ens}}}{2}} \mathcal{O}_i \quad \text{and} \quad B = \frac{2}{N_{\text{ens}}} \sum_i^{\frac{N_{\text{ens}}}{2}} B_i . \quad (9.2.5)$$

The uncertainties  $\delta_{\mathcal{O}}$  and  $\delta_B$  on these averages are determined via the standard deviation of the individual contributions, specifically given by

$$\delta_{\mathcal{O}} = \frac{2}{N_{\text{ens}}} \sigma(\mathcal{O}_i) \quad \text{and} \quad \delta_B = \frac{2}{N_{\text{ens}}} \sigma(B_i) . \quad (9.2.6)$$

With these average values, we can determine the combined signal amount  $S$  and its uncertainty  $\delta_S$  using

$$S = \mathcal{O} - B \quad \text{and} \quad \delta_S = \sqrt{\mathcal{O}^2 + B^2} . \quad (9.2.7)$$

This finally allows the definition of the ONLINEFLOW significance as



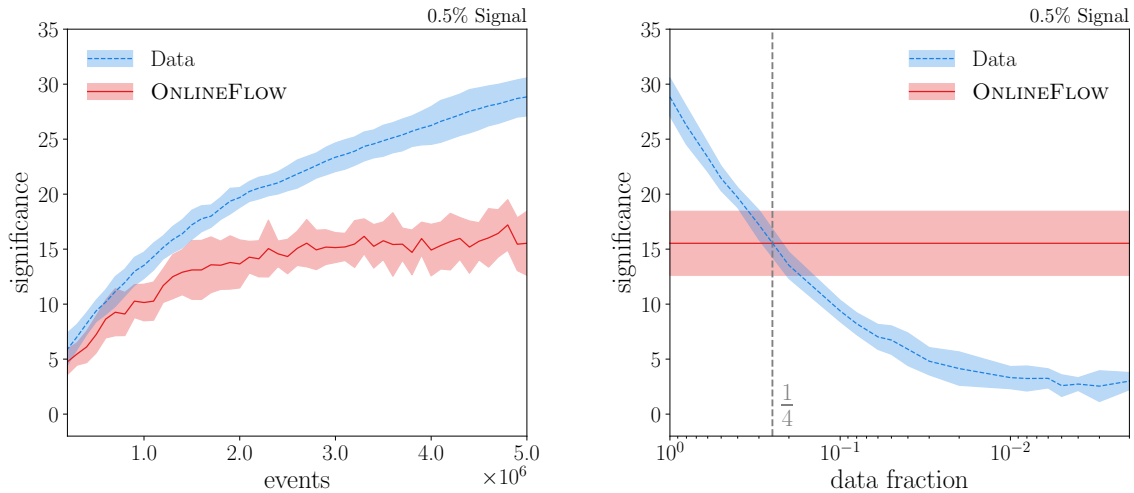


Figure 9.5: Significance obtained from the training data (blue) and from the ONLINEFLOW data (red) over the course of the online training (left) and as a function of the prescale factor (right). The shaded regions correspond to the uncertainty obtained from 10 training runs with independently generated data. The dotted line in the right panel marks the crossover point between ONLINEFLOW and training data significance. Figure taken from [4].

$$\text{significance}_{\text{ONLINEFLOW}} = \frac{S}{\sqrt{\delta_S^2 + (\sqrt{B})^2}} . \quad (9.2.8)$$

While the large number of generated samples likely makes the  $\sqrt{B}$  contribution to the uncertainty negligible, as discussed previously, we still preserve the term for consistency.

Using this significance definition we can now evaluate the performance of the ONLINEFLOW over the course of the online training. The left panel of Figure 9.5 shows the training data significance in blue and the ONLINEFLOW significance in red, as functions of the numbers of points seen during the online training. For the training data a prescale factor of  $f_{\text{pre}} = 1$  was used. The shaded envelopes around the curves show the uncertainty obtained from the standard deviation over 10 training runs on independent data sets. We can see the training significance continuously increase as more data is added. As would be expected from the significance definition in Equation (9.2.4), the significance scales approximately with the square root of the number of data points. The ONLINEFLOW significance curve is noticeably lower than the training data curve. This is expected as the information of the training data is not perfectly encoded into the network weights. Nevertheless, the ONLINEFLOW significance reaches a respectable significance and shows a scaling with the number of data points qualitatively similar to the training data curve.

This is a promising result for the application at HLT level, as it demonstrates that an online trained model can indeed be used to accurately find excesses in data.

In order to evaluate the performance of the ONLINEFLOW when applied after the HLT, we compare the ONLINEFLOW significance to the training data significance obtained for various

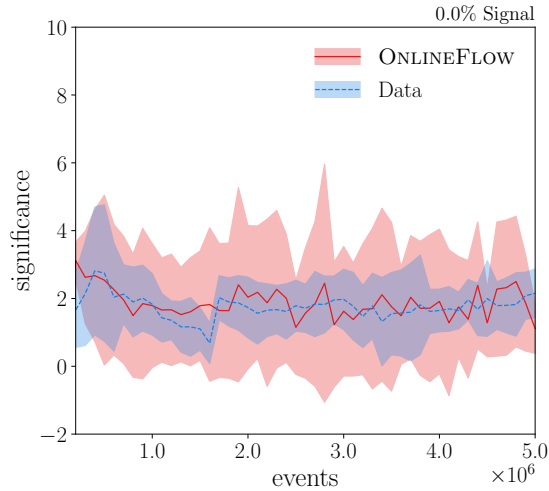


Figure 9.6: Significance obtained from the training data (blue) and from the ONLINEFLOW data (red) over the course of the online training for a data set with no signal present. Figure taken from [4].

data fractions. This comparison can be seen in the right panel of Figure 9.5. The ONLINEFLOW significance for this comparison is constant, as it does not depend on the prescale factor. The training data curve drops for an increasing prescale factor, as a larger prescale factor corresponds to less total data. The crossover between ONLINEFLOW and training data occurs at a data fraction of  $\frac{1}{4}$ , which corresponds to a prescale factor of 4.

Therefore, in this 1-dimensional example, one would benefit from the ONLINEFLOW even when it is applied after the HLT, as long as the prescale factor is larger than 4.

An important concern for the ONLINEFLOW approach is its susceptibility to creating fake accesses through imperfect training. To investigate this we repeat the previous evaluation procedure for a data set with a signal rate of  $\lambda = 0$ . The results of this test are shown in Figure 9.6. While the significance of the ONLINEFLOW fluctuates more than that of the training data, both average a significance of around 2. This indicates that the ONLINEFLOW is no more prone to finding fake bumps than the classical approach.

### 9.3 Anomaly Detection Data Set

Having demonstrated the concept of the ONLINEFLOW approach on a low-complexity data set we move to a more challenging, realistic data set. The LHC anomaly detection challenge R&D data set [34] is a well-established and researched set of events. Several anomaly detection methods have already been benchmarked on this data set, making it a perfect candidate for the ONLINEFLOW model.

The data set itself consists of di-jet background events and signal events of a heavy  $W'$  resonance, with a mass of  $m_{W'} = 3.5$  TeV. The  $W'$  decays into another two heavy particles  $X$  and  $Y$ , with masses  $m_X = 500$  GeV and  $m_Y = 100$  GeV.  $X$  and  $Y$  in turn decay into light quarks, resulting in the following decay chain,

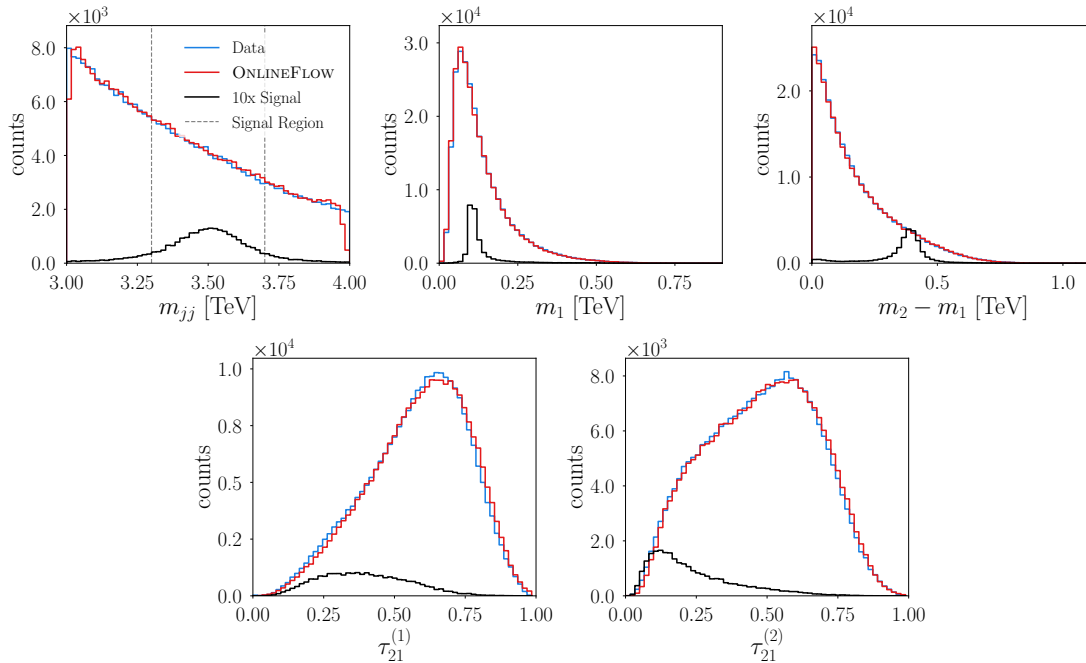


Figure 9.7: Variables used to train the ONLINEFLOW on the anomaly detection data set. The blue line corresponds to the signal and background data. The red line shows the events produced by the ONLINEFLOW model. The black curve depicts a  $10\times$  enhanced version of the signal events. The signal region used for the CWoLa approach is enclosed by the gray dashed lines in the upper left panel. Figure taken from [4].

$$W' \rightarrow X(\rightarrow qq)Y(\rightarrow qq) . \quad (9.3.1)$$

The exact composition of signal and background events is not fixed and can be adjusted. For this study, a signal rate of 1% was chosen.

The original data set was simulated in PYTHIA8 [195] and used DELPHES3.4.1 [97, 196] to approximate the detector effects. The final state jets were clustered using the anti- $k_T$  algorithm [197] with a jet radius of  $R = 1$ , implemented in FASTJET [198]. A final cut is applied to all jets, requiring at least one jet with  $p_T > 1.2$  TeV.

Training the ONLINEFLOW on the raw event data would present a significant increase in difficulty compared to the 1-dimensional data set. However, several sophisticated anomaly detection methods [184, 191, 199] have demonstrated that a small set of selected variables is sufficient to separate the signal from the background. The five variables used in these methods are the combined di-jet mass of the two leading jets  $m_{jj}$ , the invariant mass of the highest  $p_T$  jet  $m_1$ , the mass difference between the highest and second highest  $p_T$  jets  $m_1 - m_2$ , as well as the  $n$ -subjettiness [200, 201] ratios  $\tau_{21}^{(1)}$  and  $\tau_{21}^{(2)}$ . The distributions of these five variables are shown in Figure 9.7.

The data set contains 300k events in total, which were split into a training set containing 80%, a validation set containing 10%, and a test set containing the remaining 10%. The evaluation set was further supplemented with an additional 300k events with a signal rate of  $\sim 50\%$ .

This has no direct effect on the performance evaluation but does improve the presentation of the final results by smoothing out the receiver operating characteristic (ROC) and significance improvement characteristic (SIC) curves.

The training set is then used to train the ONLINEFLOW. To counterbalance the relatively low amount of training data in this example the number of iterations per buffer is increased to  $N_{\text{buffer}} = 1000$ . As the anomaly detection method used in the evaluation does not require a direct uncertainty estimation, we no longer require an ensemble of flows and instead use a single ONLINEFLOW model. We again make use of five additional noise dimensions in the flow training, increasing the input dimensionality of the flow to 10, as this was found to improve the performance. Further, the number of MADE blocks in the ONLINEFLOW architecture is doubled from 5 to 10 and the number of nodes in the FCN layers is increased from 32 to 128. Other training parameters remain identical to what was described in Section 9.2.

### Anomaly Detection Performance

We again gauge the performance of the ONLINEFLOW by how easily the signal can be extracted from the flow-generated data. To this end, we apply an anomaly detection setup to both the training and the ONLINEFLOW data. The ONLINEFLOW merely compresses the data set and does not perform any anomaly detection on its own. Therefore it should be compatible with any anomaly detection method. To avoid introducing further complexity we decided to use classification without labels (CWoLa)<sup>1</sup> [202–204], a simple-to-use and well-tested anomaly detection method.

In the CWoLa method, the data set is split into two regions containing varying relative amounts of signal. For the LHCO challenge data set this split is performed using the di-jet mass spectrum. The dashed gray lines in the top left panel of Figure 9.7 indicate how the spectrum is split into a signal region (SR) and sidebands (SB). In a realistic search, the SR would be moved along the di-jet mass to scan for anomalies, however as we are only interested in evaluating the ONLINEFLOW performance this step is omitted.

CWoLa then uses a classification network to learn the difference between SR and SB. Since the only difference between SR and SB are their signal rates, the classifier will learn to distinguish between signal and background events, even if the training data is not labeled.

The signal-like-ness of a given event can then be defined using the trained classifier as

$$R_{\text{CWoLa}}(x) = \frac{p(x|\text{SR})}{p(x|\text{SB})}, \quad (9.3.2)$$

where  $p(x|\text{SR})$  is the classifier prediction that a given event  $x$  is from the SR and  $p(x|\text{SB})$  the prediction that the event originates from the SB. Using this variable the most signal-like events can be selected.

The classifier is constructed following Reference [191] using three fully connected layers with 64 nodes. Due to the arbitrary separation into SR and SB, the two regions contain an unequal number of points. This imbalance is remedied by reweighting the data points during training. The model itself is implemented in PYTORCH and trained using a cross entropy loss and the ADAM optimizer with a learning rate of  $10^{-3}$ . The final evaluation of the classifier is performed on an ensemble of the 10 lowest validation-loss checkpoints.

<sup>1</sup>The CWoLa setup was provided by Manuel Sommerhalder

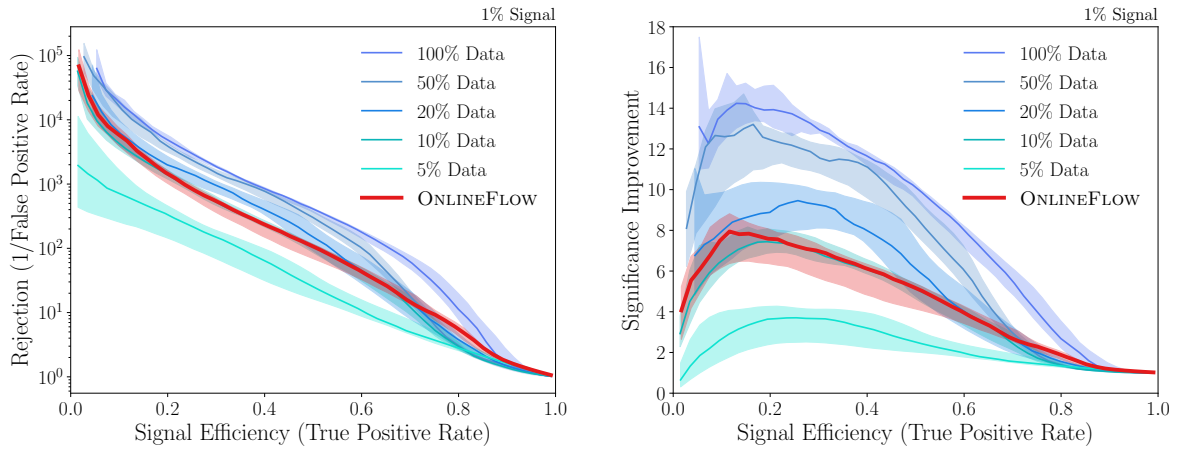


Figure 9.8: CWoLa benchmark results for the online flow (red) and the training data (shades of blue). The vertical order of the training lines on the legend corresponds to their order in the plots. Results are shown as a ROC curve (left) and a significance improvement curve (right). For the ROC and sic curves, a higher rejection and a greater significance improvement corresponds to a better performance, respectively. Figure taken from [4].

Similar to the 1-dimensional data set we establish a benchmark by applying the CWoLa approach directly to the training data used in the ONLINEFLOW training and the validation set described above. To again emulate the effect of various prescale factors we perform further trainings using 50%, 20%, 10%, and 5% of the training and validation data, corresponding to prescale factors of 2, 5, 10, and 20 respectively.

For the ONLINEFLOW we use the model to generate 500k data points to train the CWoLa classifier, and an additional 62500 samples to use as a validation set. This corresponds to the same 8 : 1 train to validation ratio used for the LHCO data. The ROC and significance improvement curve produced by the CWoLa approach are shown in Figure 9.8. From the significance improvement curve, we can see that the ONLINEFLOW data can be used to find the signal data with a significance improvement of up to 8. Further, both curves indicate that the ONLINEFLOW performs comparably to 10% of the training data, which is equivalent to a prescale factor of 10.

Therefore the ONLINEFLOW would be able to assist in finding this signal if applied at the HLT-level, and would still provide a benefit if applied after the HLT in cases where the prescale factor is 10 or greater.

## 9.4 Conclusion

We propose the use of a generative model, trained online on trigger level, to act as a scouting tool complementary to currently used trigger setups. This ONLINEFLOW could leverage its fixed storage requirement to extract and preserve information from phase-space regions with rates too large for classical data taking.

We initially demonstrated the ability of the ONLINEFLOW model to capture even low rate signal on a simple exponential falling mass distribution, where we showed that the ONLINEFLOW starts to outperform classical data taking for a prescale factor larger than 4. Further, we showed

the applicability of the ONLINEFLOW method to more complex and realistic physics data using the LHCO anomaly detection challenge data set, where the use of the ONLINEFLOW model leads to an improved anomaly detection result for prescale factors above 10.

For now, this serves as a promising proof of concept example. The practical implementation of the ONLINEFLOW setup in a real experiment will require further work to ensure the model training is sufficiently fast to keep pace with the trigger rate and can be integrated into the software and hardware frameworks.

## Chapter 10

# Summary and Outlook

Escalating MC simulation costs present a significant problem for future collider experiments, with the full simulation of calorimeter showers being the most impactful bottleneck of the simulation chain. This leads to a desire for new fast and accurate simulation methods. This thesis demonstrates how generative ML models can provide this precise simulation, as well as how generative models can be applied beyond the field of fast simulation.

An important question regarding the usage of generative models for simulation is how many points can reasonably be sampled from a generative model trained on a given data set. This question is closely linked to how well a generative model can describe the underlying distribution of the training data set. If the generative model describes the underlying distribution more accurately than the training data alone, then it can be used to *amplify* the training data set, meaning it is reasonable to sample more new points from the model than were contained in the original training data. Most generative FastSim approaches aim to generate significantly more data than they were trained on, thereby implicitly relying on such amplification behavior.

In Chapter 5, we, therefore, address this question by examining the statistical properties of a GAN trained on data sets sampled from a 1-dimensional, 2-dimensional, and 5-dimensional distribution. For the comparison between the generated data, the training data, and the true distribution we employ a quantile-based MSE metric. We find that if sufficient data is drawn from the GAN model, the newly generated data points can result in a better description of the true distribution than the training data alone. It is our assumption that this behavior can be traced back to the ability of the GAN to interpolate between individual points, leading to an overall closer approximation of the true data distribution than what is provided by the discrete training data.

This result presents an important achievement for the field of generative fast simulation as a whole, as it provides assurance that the use of generative models for simulation is not prohibited by the inherent statistical properties of the approach. Further, these results have led to subsequent work, where the amplification behavior was replicated on a more realistic physics data set [7].

Having demonstrated the ability of generative models to amplify a data set, we move to apply them to shower simulation. In Chapter 6 we present the results of using a GAN, a WGAN, and a novel BIB-AE architecture to model photon showers in the highly granular ILD [59] ECAL. To this end, we train the models using a set of showers simulated using GEANT4. We find that

all three generative models can replicate the showers produced by GEANT4 to a high level of accuracy while providing a significant speedup compared to GEANT4 on identical hardware.

Most notably the combination of the BIB-AE model with a dedicated, MMD trained Post Processor network achieves good agreement with the GEANT4 data and represents the first generative approach capable of accurately modeling the MIP peak in the cell energy spectrum. Based on this success, the concept of using a secondary network to improve the outputs of a generative model presents an interesting idea, with a range of potential applications to other generative projects. Beyond this, this work has resulted in a follow-up publication exploring the details of the BIB-AE latent space [6].

Building on the success achieved in the simulation of photon showers, we extend our efforts to the simulation of pion showers in the ILD AHCAL in Chapter 7. Compared to photon showers, pion showers feature significantly more diverse shower shapes, making this application a significantly greater challenge. Therefore, we initially demonstrate how the WGAN and BIB-AE models can be modified to learn the more complex pion data set. Most notably we show how the addition of a KDE-based latent sampling to the BIB-AE significantly increases the fidelity of the generated showers.

Comparing the relative performance of the BIB-AE and WGAN, we find that the BIB-AE produces higher quality showers at a lower generation speedup, while the WGAN struggles to correctly model the shapes of the showers, but offers a greater generation speedup factor. We further compare the particle energy reconstructed by the PANDORAPFA [80] from the generated showers. We observe that both generative models show noticeable deviations from GEANT4 after the reconstruction is applied, however, the overall agreement with GEANT4 remains reasonable.

On the one hand, these results demonstrate that a trade-off between fidelity and sampling speed exists for generative models, allowing individual applications to choose which factor to prioritize. On the other hand, we see that even a less accurate model can still model important features like the reconstructed particle energy with reasonable precision

In Chapter 8 we present the first training of a generative model on real measurement data in particle physics. We use a BIB-AE model trained directly on testbeam data recorded with a prototype of the ILD AHCAL, and compare the agreement between the BIB-AE and the data and between a GEANT4 simulation and the data. This comparison shows that a generative model trained directly on data can reach an overall comparable agreement to that reached by GEANT4, while providing a speedup of up to a factor of 50 .

With this, we have laid the groundwork for a potential new class of simulators, trained and optimized directly on measurement data, that could not only lead to an increase in computational speed, but also to an improvement of the precision of the simulation.

Chapter 9 demonstrates the use of generative models in applications other than fast simulation. In modern collider experiments, the collision rate is too high to record every event, and instead, triggers are used to select only interesting events for storage. This results in a large fraction of events being discarded. To address this, we introduce ONLINEFLOW, a generative model designed to be trained online at the trigger level. This model can learn information about events that are discarded under the current trigger scheme, without having to store additional data. Once the model is trained, the events generated by the models can be analyzed in an offline setting, in order to search for interesting physics in the otherwise discarded regions.

We demonstrate the ability of the ONLINEFLOW model to act as such a scouting tool in two proof-of-concept examples, using an emulated invariant mass spectrum and the LHCO anomaly



detection challenge data set, respectively. Using these two data sets we show that the ONLINEFLOW can learn a set sufficiently accurate to extract a low-rate signal from the ONLINEFLOW generated events.

Generative models present a promising method for dealing with data-related problems in particle physics, offering new approaches for both data simulation and data storage. We were able to demonstrate the usability of generative models for the simulation of highly granular calorimeter showers, however, this presents only a first step toward generative detector simulation. Several challenges will have to be addressed first. For one, the models presented can only produce showers of pions that hit the calorimeter at one specific point, and under one specific angle. In order to allow for practical application in a simulation chain, the models will have to be adapted to allow for the simulation of multiple impact points and angles. Further, additional work will be required to integrate the generative models into a simulation chain in a way that does not impede the performance of the model. The statistical properties of generative models are another largely unexplored topic. While we have shown that a generative model can amplify a data set, several questions remain, notably in regard to the uncertainty of the generated data. Work has been done using Bayesian neural networks to estimate the uncertainties of generative models [41], however, extending these principles to our high granularity shower simulation is highly non-trivial. A further point to address will be the training times of the generative models. For now, we have focused on the time required to simulate individual showers and assumed that the high demand for MC data will make the training time a negligible overhead. However, this is not guaranteed to be the case for more complex models and more intricate detector geometries. Should training times become problematic, advanced training methods such as transfer learning [205] or meta learning [206] offer a promising avenue to reduce the time required to train new networks. The idea of online trained generative models as scouting tools shows promise, however, the technical challenges of training a generative network on trigger level are rather daunting and will require further research. Finally, the successful training of a generative network on real data presents an exciting opportunity, the exact limitations, and abilities of which remain to be explored in more detail.



# Acknowledgments

The author was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2121 “Quantum Universe” – 390833306. This research was supported in part through the Maxwell computational resources operated at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

With that out of the way, this thesis represents the culmination of three of the genuinely most enjoyable years of my life (and that is saying a lot, considering two of those years were under Covid-included lockdown). None of this would have been possible without the help of a multitude of amazing people, and while I might not be able to thank all of them, I sure will try.

First of all, I would like to thank Prof. Dr. Gregor Kasieczka and Dr. Frank Gaede, for the great supervision, the interesting discussion, readily available assistance, truly welcoming attitude, and ever-welcome challenges they provided me with over the course of my doctoral studies.

Similarly, I would like to extend my sincerest gratitude to Dr. Freya Blekman, Prof. Dr. Marcus Brüggem, and Prof. Dr. Günter Sigl for being part of my examination committee.

My deepest thanks go out to Katja Krüger, Karim El Morabit, Louis Moureaux, Ramon Winterhalder, Peter McKeown, Manuel Sommerhalder, and Sebastian Bieringer for their endless patience and willingness to take time out of their already packed schedules to proofread this work.

Further, I would like to thank my former Heidelberg supervisor Tilman Plehn, who not only guided me through my Bachelor's and Master's but also for putting me in the pursuit of ML in HEP, which ultimately led to this thesis.

I am incredibly thankful to the collaborators I had the pleasure to work with over the years, Sebastian Bieringer, Erik Buhmann, Anja Butter, Engin Eren, Daniel Hundhausen, Daniel Heuchel, William Korcari, Anatolii Korol, Katja Krüger, Benjamin Nachman, Peter McKeown, Tilman Plehn, Lennart Rustige, and Ramon Winterhalder. Your insight and willingness to discuss was truly invaluable.

Similarly, I would like to thank the students, who I had the honor of supervising – Daniel Hundhausen, Tore von Schwartz, Imahn Shekhzadeh, and Noah Tettenborn – for ensuring I never got too complacent.

In the same vein I would like to thank the current and former members of the Kasieczka UHH CMS group and the FTX SFT group (ordered according to the likely incorrect office plan in my Head), Karla Peña, Melanie Eich, Jörg Schindler, Lisa Benato, Karim El Morabit, Philipp Rincke, Nils Gerber, Lukas Judith, Christian Elsässer, Julia Heiken, Louis Moureaux, Tobias Quadfasel, Manuel Sommerhalder, Sven Bollweg, Lennart Kämmler, Veronika Kinsvater, Parada Prangchaikul, Nana Marie Werther, Noah Tettenborn, Imahn Shekhzadeh, Daniel Hundhouse,

Malte Jacobsen, Moritz Wohlstein, Tore von Schwartz, Sebastian Bieringer, William Korcari, Erik Buhmann, Lennart Rustige, Engin Eren, Peter McKeown, Thomas Madlener, Anatolii Korol, Francisca Wolf, Jamal Slim, Finn Johannsen, and Mareike Meyer. I cannot even begin to describe the amazing ways you make my time in Hamburg worth every second, so I will briefly list things: Laser Tag, weddings, canoe trips, physics questions, social evenings, Magic: The Gathering, lunches at CFEL, nights out, fondue, lunches at the canteen, cake-related compliments, and much more.

In a similar vein, a special thanks also to the owners of Trollskull Manor and the crew of the ~~Wolpertinger Manticore~~ ~~Tisiphone~~ Manticore, who provided a much-needed reprieve from the stress of work, by replacing it with a whole different kind of stress. You made the dark periods during the lockdown so much brighter.

Finally, I want to express my deepest gratitude to my parents and family, not just for supporting me during the pursuit of my dream, but, maybe even more importantly, for making sure I knew just how unconditional this support is.

# Bibliography

- [1] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn 2021 **GAN-  
plifying event samples** *SciPost Phys.* **10** 139. e-Print: 2008.06545 doi:  
10.21468/SciPostPhys.10.6.139
- [2] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger  
2021 **Getting High: High Fidelity Simulation of High Granularity Calorime-  
ters with High Speed** *Comput. Softw. Big Sci.* **5** 13. e-Print: 2005.05334 doi:  
10.1007/s41781-021-00056-0
- [3] 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
- [4] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, T. Plehn, D. Shih and R. Win-  
terhalder 2022 **Ephemeral Learning – Augmenting Triggers with Online-Trained  
Normalizing Flows** e-Print: 2202.09375
- [5] S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman and D. Shih 2020 **DCTR-  
GAN: Improving the Precision of Generative Models with Reweighting** *JINST*  
**15** P11004. e-Print: 2009.03796 doi: 10.1088/1748-0221/15/11/P11004
- [6] 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
- [7] 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
- [8] S. Weinberg 1967 **A Model of Leptons** *Phys. Rev. Lett.* **19**(21) 1264 doi:  
10.1103/PhysRevLett.19.1264
- [9] S. L. Glashow 1961 **Partial-symmetries of weak interactions** *Nuclear Physics* **22** 579  
doi: 10.1016/0029-5582(61)90469-2
- [10] D. J. Gross and F. Wilczek edited by J. C. Taylor 1973 **Ultraviolet Behavior of Non-  
abelian Gauge Theories** *Phys. Rev. Lett.* **30** 1343 doi: 10.1103/PhysRevLett.30.1343

- 
- [11] H. D. Politzer edited by J. C. Taylor 1973 **Reliable Perturbative Results for Strong Interactions?** *Phys. Rev. Lett.* **30** 1346 doi: 10.1103/PhysRevLett.30.1346
- [12] F. Englert and R. Brout edited by J. C. Taylor 1964 **Broken Symmetry and the Mass of Gauge Vector Mesons** *Phys. Rev. Lett.* **13** 321 doi: 10.1103/PhysRevLett.13.321
- [13] P. W. Higgs 1964 **Broken symmetries, massless particles and gauge fields** *Phys. Lett.* **12** 132 doi: 10.1016/0031-9163(64)91136-9
- [14] P. W. Higgs edited by J. C. Taylor 1964 **Broken Symmetries and the Masses of Gauge Bosons** *Phys. Rev. Lett.* **13** 508 doi: 10.1103/PhysRevLett.13.508
- [15] The ATLAS collaboration 2012 **Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC** *Phys. Lett. B* **716** 1. e-Print: 1207.7214 doi: 10.1016/j.physletb.2012.08.020
- [16] The CMS collaboration 2012 **Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC** *Phys. Lett. B* **716** 30. e-Print: 1207.7235 doi: 10.1016/j.physletb.2012.08.021
- [17] E. Corbelli and P. Salucci 2000 **The Extended Rotation Curve and the Dark Matter Halo of M33** *Mon. Not. Roy. Astron. Soc.* **311** 441. e-Print: astro-ph/9909252 doi: 10.1046/j.1365-8711.2000.03075.x
- [18] D. Clowe, M. Bradac, A. H. Gonzalez, M. Markevitch, S. W. Randall, C. Jones and D. Zaritsky 2006 **A direct empirical proof of the existence of dark matter** *Astrophys. J. Lett.* **648** L109. e-Print: astro-ph/0608407 doi: 10.1086/508162
- [19] G. Hinshaw *et al.* (WMAP) 2009 **Five-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Data Processing, Sky Maps, and Basic Results** *Astrophys. J. Suppl.* **180** 225. e-Print: 0803.0732 doi: 10.1088/0067-0049/180/2/225
- [20] P. A. R. Ade *et al.* (Planck) 2016 **Planck 2015 results. XIII. Cosmological parameters** *Astron. Astrophys.* **594** A13. e-Print: 1502.01589 doi: 10.1051/0004-6361/201525830
- [21] S. F. King 2004 **Neutrino mass models** *Rept. Prog. Phys.* **67** 107. e-Print: hep-ph/0310204 doi: 10.1088/0034-4885/67/2/R01
- [22] G. Ciezarek, M. Franco Sevilla, B. Hamilton, R. Kowalewski, T. Kuhr, V. Lüth and Y. Sato 2017 **A Challenge to Lepton Universality in B Meson Decays** *Nature* **546** 227. e-Print: 1703.01766 doi: 10.1038/nature22346
- [23] The LHCb collaboration 2014 **Test of lepton universality using  $B^+ \rightarrow K^+ \ell^+ \ell^-$  decays** *Phys. Rev. Lett.* **113** 151601. e-Print: 1406.6482 doi: 10.1103/PhysRevLett.113.151601
- [24] L. Evans and P. Bryant 2008 **LHC Machine** *JINST* **3** S08001 doi: 10.1088/1748-0221/3/08/S08001
- [25] 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

- [26] 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
- [27] T. K. Charles *et al.* (CLICdp, CLIC) edited by P. N. Burrows, N. Catalan Lasheras, L. Linssen, M. Petrič, A. Robson, D. Schulte, E. Sicking and S. Stapnes 2018 **The Compact Linear Collider (CLIC) - 2018 Summary Report 2/2018**. e-Print: 1812.06018 doi: 10.23731/CYRM-2018-002
- [28] R. Jansky 2015 **The ATLAS Fast Monte Carlo Production Chain Project** *J. Phys. Conf. Ser.* **664** 072024 doi: 10.1088/1742-6596/664/7/072024
- [29] S. Agostinelli *et al.* (GEANT4) 2003 **GEANT4—a simulation toolkit** *Nucl. Instrum. Meth. A* **506** 250 doi: 10.1016/S0168-9002(03)01368-8
- [30] The ATLAS collaboration 2020 **ATLAS HL-LHC Computing Conceptual Design Report** Tech. rep. CERN Geneva URL: <https://cds.cern.ch/record/2729668>
- [31] The CMS collaboration 2018 **Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV** *JINST* **13** P05011. e-Print: 1712.07158 doi: 10.1088/1748-0221/13/05/P05011
- [32] D. Guest, J. Collado, P. Baldi, S.-C. Hsu, G. Urban and D. Whiteson 2016 **Jet Flavor Classification in High-Energy Physics with Deep Neural Networks** *Phys. Rev. D* **94** 112002. e-Print: 1607.08633 doi: 10.1103/PhysRevD.94.112002
- [33] A. Butter *et al.* edited by G. Kasieczka and T. Plehn 2019 **The Machine Learning landscape of top taggers** *SciPost Phys.* **7** 014. e-Print: 1902.09914 doi: 10.21468/SciPostPhys.7.1.014
- [34] G. Kasieczka *et al.* 2021 **The LHC Olympics 2020 a community challenge for anomaly detection in high energy physics** *Rept. Prog. Phys.* **84** 124201. e-Print: 2101.08320 doi: 10.1088/1361-6633/ac36b9
- [35] A. Butter, T. Plehn and R. Winterhalder 2019 **How to GAN LHC Events** *SciPost Phys.* **7** 075. e-Print: 1907.03764 doi: 10.21468/SciPostPhys.7.6.075
- [36] B. Hashemi, N. Amin, K. Datta, D. Olivito and M. Pierini 2019 **LHC analysis-specific datasets with Generative Adversarial Networks** e-Print: 1901.05282
- [37] R. Di Sipio, M. Fauci Giannelli, S. Ketabchi Haghighat and S. Palazzo 2019 **DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC** *JHEP* **08** 110. e-Print: 1903.02433 doi: 10.1007/JHEP08(2019)110
- [38] J. Arjona Martínez, T. Q. Nguyen, M. Pierini, M. Spiropulu and J.-R. Vlimant 2020 **Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description** *J. Phys. Conf. Ser.* **1525** 012081. e-Print: 1912.02748 doi: 10.1088/1742-6596/1525/1/012081

- 
- [39] Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, P. Ambrozewicz, F. Hauenstein, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco and Y. Li 2021 **Simulation of Electron-Proton Scattering Events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)** *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* edited by Z.-H. Zhou (IJCAI) p 2126. e-Print: 2001.11103 doi: 10.24963/ijcai.2021/293
- [40] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri and R. Verheyen 2021 **Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer** *Nature Commun.* **12** 2985. e-Print: 1901.00875 doi: 10.1038/s41467-021-22616-z
- [41] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot and S. Vent 2021 **Generative Networks for Precision Enthusiasts** e-Print: 2110.13632
- [42] J. W. Monk 2018 **Deep Learning as a Parton Shower** *JHEP* **12** 021. e-Print: 1807.03685 doi: 10.1007/JHEP12(2018)021
- [43] K. Dohi 2020 **Variational Autoencoders for Jet Simulation** e-Print: 2009.04842
- [44] B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J. Duarte, R. Kansal, J.-R. Vlimant and D. Gunopulos 2021 **Sparse Data Generation for Particle-Based Simulation of Hadronic Jets in the LHC** *38th International Conference on Machine Learning*. e-Print: 2109.15197
- [45] S. Tsan, R. Kansal, A. Aportela, D. Diaz, J. Duarte, S. Krishna, F. Mokhtar, J.-R. Vlimant and M. Pierini 2021 **Particle Graph Autoencoders and Differentiable, Learned Energy Mover's Distance** *35th Conference on Neural Information Processing Systems*. e-Print: 2111.12849
- [46] 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
- [47] 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
- [48] 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
- [49] 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



- [50] 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
- [51] S. Vallecorsa 2018 **Generative models for fast simulation** *J. Phys. Conf. Ser.* **1085** 022005 doi: 10.1088/1742-6596/1085/2/022005
- [52] 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
- [53] K. Deja, T. Trzcinski and L. Graczykowski edited by A. Forti, L. Betev, M. Litmaath, O. Smirnova and P. Hristov 2019 **Generative models for fast cluster simulations in the TPC for the ALICE experiment** *EPJ Web Conf.* **214** 06003 doi: 10.1051/epjconf/201921406003
- [54] 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
- [55] A. Hariri, D. Dyachkova and S. Gleyzer 2021 **Graph Generative Models for Fast Detector Simulations in High Energy Physics** e-Print: 2104.01725
- [56] V. Chekalina, E. Orlova, F. Ratnikov, D. Ulyanov, A. Ustyuzhanin and E. Zakharov edited by A. Forti, L. Betev, M. Litmaath, O. Smirnova and P. Hristov 2019 **Generative Models for Fast Calorimeter Simulation: the LHCb case** *EPJ Web Conf.* **214** 02034. e-Print: 1812.01319 doi: 10.1051/epjconf/201921402034
- [57] The ATLAS collaboration 2018 **Deep generative models for fast shower simulation in ATLAS** Tech. rep. CERN Geneva URL: <http://cds.cern.ch/record/2630433>
- [58] K. Cranmer, J. Pavez and G. Louppe 2015 **Approximating Likelihood Ratios with Calibrated Discriminative Classifiers** e-Print: 1506.02169
- [59] The ILD Concept Group edited by T. Behnke *et al.* 2020 **International Large Detector: Interim Design Report** e-Print: 2003.01116
- [60] B. Nachman and J. Thaler 2021 **Learning from many collider events at once** *Phys. Rev. D* **103** 116013. e-Print: 2101.07263 doi: 10.1103/PhysRevD.103.116013
- [61] The CMS collaboration 2022 **Denoising Convolutional Networks to Accelerate Detector Simulation** *20th International Workshop on Advanced Computing and Analysis Techniques in Physics Research: AI Decoded - Towards Sustainable, Diverse, Performant and Effective Scientific Computing.* e-Print: 2202.05320 doi: 10.2172/1835859
- [62] The CALICE collaboration 2022 **The CALICE collaboration** [Online; accessed 22-September-2022] URL: <https://twiki.cern.ch/twiki/bin/view/CALICE/CaliceCollaboration>

- 
- [63] The CALICE collaboration 2010 **Construction and Commissioning of the CALICE Analog Hadron Calorimeter Prototype** *JINST* **5** P05004. e-Print: 1003.2662 doi: 10.1088/1748-0221/5/05/P05004
- [64] A. Pich 2012 **The Standard Model of Electroweak Interactions** *2010 European School of High Energy Physics* p 1. e-Print: 1201.0537
- [65] MissMJ, Cush 2019 **Standard Model of Elementary Particles** [Online; accessed 10-September-2022] URL: [https://upload.wikimedia.org/wikipedia/commons/0/00/Standard\\_Model\\_of\\_Elementary\\_Particles.svg](https://upload.wikimedia.org/wikipedia/commons/0/00/Standard_Model_of_Elementary_Particles.svg)
- [66] M. Thomson 2013 **Modern particle physics** (New York: Cambridge University Press) ISBN 978-1-107-03426-6
- [67] The LHCb collaboration 2014 **Observation of the resonant character of the  $Z(4430)^-$  state** *Phys. Rev. Lett.* **112** 222002. e-Print: 1404.1903 doi: 10.1103/PhysRevLett.112.222002
- [68] The LHCb collaboration 2015 **Observation of  $J/\psi p$  Resonances Consistent with Pentaquark States in  $\Lambda_b^0 \rightarrow J/\psi K^- p$  Decays** *Phys. Rev. Lett.* **115** 072001. e-Print: 1507.03414 doi: 10.1103/PhysRevLett.115.072001
- [69] S. N. Ahmed *et al.* (SNO) 2004 **Measurement of the total active B-8 solar neutrino flux at the Sudbury Neutrino Observatory with enhanced neutral current sensitivity** *Phys. Rev. Lett.* **92** 181301. e-Print: nucl-ex/0309004 doi: 10.1103/PhysRevLett.92.181301
- [70] M. Gell-Mann, P. Ramond and R. Slansky 1979 **Complex Spinors and Unified Theories** *Conf. Proc. C* **790927** 315. e-Print: 1306.4669
- [71] P. Langacker 2009 **The Physics of Heavy  $Z'$  Gauge Bosons** *Rev. Mod. Phys.* **81** 1199. e-Print: 0801.1345 doi: 10.1103/RevModPhys.81.1199
- [72] M. Bauer, S. Diefenbacher, T. Plehn, M. Russell and D. A. Camargo 2018 **Dark Matter in Anomaly-Free Gauge Extensions** *SciPost Phys.* **5** 036. e-Print: 1805.01904 doi: 10.21468/SciPostPhys.5.4.036
- [73] The ATLAS collaboration 2008 **The ATLAS Experiment at the CERN Large Hadron Collider** *JINST* **3** S08003 doi: 10.1088/1748-0221/3/08/S08003
- [74] The CMS collaboration 2008 **The CMS Experiment at the CERN LHC** *JINST* **3** S08004 doi: 10.1088/1748-0221/3/08/S08004
- [75] The LHCb collaboration 2008 **The LHCb Detector at the LHC** *JINST* **3** S08005 doi: 10.1088/1748-0221/3/08/S08005
- [76] The ALICE collaboration 2008 **The ALICE experiment at the CERN LHC** *JINST* **3** S08002 doi: 10.1088/1748-0221/3/08/S08002
- [77] The CMS collaboration 2017 **Particle-flow reconstruction and global event description with the CMS detector** *Journal of Instrumentation* **12** P10003. e-Print: 1706.04965 doi: 10.1088/1748-0221/12/10/P10003

- [78] A. D. Martin, W. J. Stirling, R. S. Thorne and G. Watt 2009 **Parton distributions for the LHC** *Eur. Phys. J. C* **63** 189. e-Print: 0901.0002 doi: 10.1140/epjc/s10052-009-1072-5
- [79] H. Abramowicz *et al.* edited by T. Behnke, J. E. Brau, P. N. Burrows, J. Fuster, M. Peskin, M. Stanitzki, Y. Sugimoto, S. Yamada and H. Yamamoto 2013 **The International Linear Collider Technical Design Report - Volume 4: Detectors** e-Print: 1306.6329
- [80] J. S. Marshall and M. A. Thomson 2015 **The Pandora software development kit for pattern recognition** *The European Physical Journal C* **75**. e-Print: 1506.05348 doi: 10.1140/epjc/s10052-015-3659-3
- [81] M. Aicheler, P. N. Burrows, N. Catalan Lasheras, R. Corsini, M. Draper, J. Osborne, D. Schulte, S. Stapnes and M. J. Stuart (CLIC accelerator) edited by M. Aicheler, P. N. Burrows, N. Catalan Lasheras, R. Corsini, M. Draper, J. Osborne, D. Schulte, S. Stapnes and M. J. Stuart 2018 **The Compact Linear Collider (CLIC) - Project Implementation Plan 4/2018**. e-Print: 1903.08655 doi: 10.23731/CYRM-2018-004
- [82] S. Stapnes 2019 **The Compact Linear Collider** *Nature Rev. Phys.* **1** 235 doi: 10.1038/s42254-019-0051-5
- [83] H. Baer *et al.* edited by H. Baer *et al.* 2013 **The International Linear Collider Technical Design Report - Volume 2: Physics** e-Print: 1306.6352
- [84] Benchmarking Working Group 2017 **HEP-SPEC06 Benchmark** [Online; accessed 16-September-2022] URL: <https://w3.hepix.org/benchmarking>
- [85] K. Hagiwara *et al.* (Particle Data Group) 2002 **Review of Particle Properties** *Phys. Rev. D* **66**(1) 010001 doi: 10.1103/PhysRevD.66.010001
- [86] J. Beringer *et al.* (Particle Data Group) 2012 **Review of Particle Physics** *Phys. Rev. D* **86**(1) 010001 doi: 10.1103/PhysRevD.86.010001
- [87] P. L. Rocca and F. Riggi 2011 **The Use of Avalanche Photodiodes in High Energy Electromagnetic Calorimetry** (Rijeka: IntechOpen) chap 12 ISBN 9789533071633 doi: 10.5772/14574
- [88] I. C. Brock and T. Schorner-Sadenius 2011 **Physics at the terascale** (Weinheim: Wiley) ISBN 9783527634965 doi: 10.1002/9783527634965
- [89] R. L. Workman and Others (Particle Data Group) 2022 **Review of Particle Physics** *PTEP* **2022** 083C01 doi: 10.1093/ptep/ptac097
- [90] S. Lee, M. Livan and R. Wigmans 2018 **On the limits of the hadronic energy resolution of calorimeters** *Nucl. Instrum. Meth. A* **882** 148. e-Print: 1710.10535 doi: 10.1016/j.nima.2017.10.087
- [91] The CMS collaboration 2015 **The CMS Electromagnetic Calorimeter: overview, lessons learned during Run 1 and future projections** *Journal of Physics: Conference Series* **587** 012001 doi: 10.1088/1742-6596/587/1/012001

- 
- [92] The CMS collaboration 2017 **The CMS HGCal detector for HL-LHC upgrade 5th Large Hadron Collider Physics Conference.** e-Print: 1708.08234 doi: 10.48550/arXiv.1708.08234
- [93] C. W. Fabjan and F. Gianotti 2003 **Calorimetry for particle physics** *Rev. Mod. Phys.* **75** 1243 doi: 10.1103/RevModPhys.75.1243
- [94] D. H. Wright and M. H. Kelsey 2015 **The Geant4 Bertini Cascade** *Nucl. Instrum. Meth. A* **804** 175 doi: 10.1016/j.nima.2015.09.058
- [95] G. Folger and J. P. Wellisch 2003 **String parton models in GEANT4** *eConf C0303241* MOMT007. e-Print: nucl-th/0306007 doi: 10.48550/arXiv.nucl-th/0306007
- [96] B. Andersson, G. Gustafson and B. Nilsson-Almqvist 1987 **A Model for Low  $p(t)$  Hadronic Reactions, with Generalizations to Hadron - Nucleus and Nucleus-Nucleus Collisions** *Nucl. Phys. B* **281** 289 doi: 10.1016/0550-3213(87)90257-4
- [97] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens and M. Selvaggi (DELPHES 3) 2014 **DELPHES 3, A modular framework for fast simulation of a generic collider experiment** *JHEP* **02** 057. e-Print: 1307.6346 doi: 10.1007/JHEP02(2014)057
- [98] 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 doi: 10.48550/arXiv.hep-ex/0001020
- [99] G. Grindhammer, M. Rudowicz and S. Peters 1990 **The Fast Simulation of Electromagnetic and Hadronic Showers** *Nucl. Instrum. Meth. A* **290** 469 doi: 10.1016/0168-9002(90)90566-0
- [100] 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
- [101] S. Glazov edited by Y. Wang 2011 **Fast simulation of showers in the H1 SpaCal calorimeter** *J. Phys. Conf. Ser.* **293** 012024 doi: 10.1088/1742-6596/293/1/012024
- [102] The CMS collaboration 2017 **The Phase-2 Upgrade of the CMS Endcap Calorimeter** Tech. rep. CERN Geneva doi: 10.17181/CERN.IV8M.1JY2 URL: <https://cds.cern.ch/record/2293646>
- [103] The CALICE collaboration 2008 **Design and Electronics Commissioning of the Physics Prototype of a Si-W Electromagnetic Calorimeter for the International Linear Collider** *JINST* **3** P08001. e-Print: 0805.4833 doi: 10.1088/1748-0221/3/08/P08001
- [104] Y. Niu *et al.* 2020 **Design of Sc-ECAL prototype for CEPC and performance of first two layers** *JINST* **15** C05036. e-Print: 2002.01809 doi: 10.1088/1748-0221/15/05/C05036

- [105] The CALICE collaboration 2013 **Validation of GEANT4 Monte Carlo Models with a Highly Granular Scintillator-Steel Hadron Calorimeter** *JINST* **8** 07005. e-Print: 1306.3037 doi: 10.1088/1748-0221/8/07/P07005
- [106] I. Goodfellow, Y. Bengio and A. Courville 2016 **Deep Learning** (MIT Press) ISBN 9780262035613 URL: <http://www.deeplearningbook.org>
- [107] S. Ruder 2016 **An overview of gradient descent optimization algorithms** e-Print: 1609.04747 doi: 10.48550/arxiv.1609.04747
- [108] D. P. Kingma and J. Ba 2015 **Adam: A Method for Stochastic Optimization** *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* edited by Y. Bengio and Y. LeCun. e-Print: 1412.6980 doi: 10.48550/arXiv.1412.6980
- [109] D. M. Hawkins 2004 **The Problem of Overfitting** *Journal of Chemical Information and Computer Sciences* **44** 1 pMID: 14741005 doi: 10.1021/ci0342472
- [110] M. Belkin, D. Hsu, S. Ma and S. Mandal 2019 **Reconciling modern machine-learning practice and the classical bias–variance trade-off** *Proceedings of the National Academy of Sciences* **116** 15849. e-Print: 1812.11118 doi: 10.1073/pnas.1903070116
- [111] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov 2014 **Dropout: A Simple Way to Prevent Neural Networks from Overfitting** *Journal of Machine Learning Research* **15** 1929
- [112] S. Ioffe and C. Szegedy 2015 **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift** *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research vol 37)* edited by F. Bach and D. Blei (Lille, France: PMLR) p 448. e-Print: 1502.03167 doi: 10.48550/arXiv.1502.03167
- [113] A. F. Agarap 2018 **Deep Learning using Rectified Linear Units (ReLU)** e-Print: 1803.08375 doi: 10.48550/arxiv.1803.08375
- [114] 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](http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf)
- [115] D. Clevert, T. Unterthiner and S. Hochreiter 2016 **Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)** *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* edited by Y. Bengio and Y. LeCun. e-Print: 1511.07289 doi: 10.48550/arXiv.1511.07289
- [116] L. Deng 2012 **The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]** *IEEE Signal Processing Magazine* **29** 141–142 doi: 10.1109/MSP.2012.2211477

- 
- [117] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen and I. Sutskever 2021 **Zero-Shot Text-to-Image Generation** *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research vol 139)* edited by M. Meila and T. Zhang (PMLR) p 8821. e-Print: 2102.12092 doi: 10.48550/arXiv.2102.12092
- [118] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio 2020 **Generative Adversarial Networks** *Commun. ACM* **63** 139. e-Print: 1406.2661 doi: 10.1145/3422622
- [119] M. Arjovsky, S. Chintala and L. Bottou 2017 **Wasserstein GAN** e-Print: 1701.07875 doi: 10.48550/arxiv.1701.07875
- [120] C. Villani 2009 **The Wasserstein distances** (Berlin, Heidelberg: Springer Berlin Heidelberg) p 93 ISBN 9783540710509 doi: 10.1007/978-3-540-71050-9\_6
- [121] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin and A. Courville 2017 **Improved Training of Wasserstein GANs** *Proceedings of the 31st International Conference on Neural Information Processing Systems NIPS'17* (Red Hook, NY, USA: Curran Associates Inc.) p 5769 ISBN 9781510860964. e-Print: 1704.00028
- [122] L. Theis, W. Shi, A. Cunningham and F. Huszár 2017 **Lossy Image Compression with Compressive Autoencoders** *International Conference on Learning Representations (ICLR 2017)* p 253. e-Print: 1703.00395 doi: 10.17863/CAM.51995
- [123] Z. Cheng, H. Sun, M. Takeuchi and J. Katto 2018 **Deep Convolutional AutoEncoder-based Lossy Image Compression** *2018 Picture Coding Symposium (PCS)* p 253. e-Print: 1804.09535 doi: 10.1109/PCS.2018.8456308
- [124] J. Liu, S. Di, K. Zhao, S. Jin, D. Tao, X. Liang, Z. Chen and F. Cappello 2021 **Exploring Autoencoder-based Error-bounded Compression for Scientific Data** *2021 IEEE International Conference on Cluster Computing (CLUSTER)* p 294. e-Print: 2105.11730 doi: 10.1109/Cluster48925.2021.00034
- [125] T. Heimel, G. Kasieczka, T. Plehn and J. M. Thompson 2019 **QCD or What?** *SciPost Phys.* **6** 030. e-Print: 1808.08979 doi: 10.21468/SciPostPhys.6.3.030
- [126] T. Finke, M. Krämer, A. Morandini, A. Mück and I. Oleksiyuk 2021 **Autoencoders for unsupervised anomaly detection in high energy physics** *JHEP* **06** 161. e-Print: 2104.09051 doi: 10.1007/JHEP06(2021)161
- [127] B. M. Dillon, T. Plehn, C. Sauer and P. Sorrenson 2021 **Better Latent Spaces for Better Autoencoders** *SciPost Phys.* **11** 061. e-Print: 2104.08291 doi: 10.21468/SciPostPhys.11.3.061
- [128] D. P. Kingma and M. Welling 2013 **Auto-Encoding Variational Bayes** e-Print: 1312.6114 doi: 10.48550/arxiv.1312.6114
- [129] J. Altosaar 2016 **Tutorial - What is a Variational Autoencoder?** Zenodo doi: 10.5281/zenodo.4462916

- [130] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow and B. Frey 2015 **Adversarial Autoencoders** e-Print: 1511.05644 doi: 10.48550/arxiv.1511.05644
- [131] S. Voloshynovskiy, M. Kondah, S. Rezaeifar, O. Taran, T. Holotyak and D. J. Rezende 2019 **Information bottleneck through variational glasses** e-Print: 1912.00830 doi: 10.48550/arxiv.1912.00830
- [132] N. Tishby, F. C. Pereira and W. Bialek 2000 **The information bottleneck method** e-Print: physics/0004057 doi: 10.48550/arxiv.PHYSICS/0004057
- [133] N. Tishby and N. Zaslavsky 2015 **Deep learning and the information bottleneck principle** *2015 IEEE Information Theory Workshop (ITW)* p 1. e-Print: 1503.02406 doi: 10.1109/ITW.2015.7133169
- [134] T. M. Cover and J. A. Thomas 2005 **Information Theory and Statistics** (John Wiley & Sons, Ltd) chap 11, p 347 ISBN 9780471748823 doi: 10.1002/047174882X.ch11
- [135] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf and A. Smola 2006 **A Kernel Method for the Two-Sample-Problem** *Advances in Neural Information Processing Systems* vol 19 edited by B. Schölkopf, J. Platt and T. Hoffman (MIT Press). e-Print: 0805.2368 doi: 10.48550/arXiv.0805.2368
- [136] S. Bond-Taylor, A. Leach, Y. Long and C. G. Willcocks 2021 **Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models** *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1 doi: 10.1109/tpami.2021.3116668
- [137] 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. e-Print: 1908.09257 doi: 10.1109/tpami.2020.2992934
- [138] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed and B. Lakshminarayanan 2021 **Normalizing Flows for Probabilistic Modeling and Inference** *Journal of Machine Learning Research* **22** 1. e-Print: 1912.02762 doi: 10.48550/arxiv.1912.02762
- [139] W. Rudin and the Tata McGraw-Hill Publishing Company 1987 **Real and Complex Analysis** Higher Mathematics Series (McGraw-Hill Education) ISBN 9780070542341
- [140] V. Bogachev 2010 **Measure Theory** (Springer Berlin Heidelberg) ISBN 9783540824640
- [141] L. Dinh, D. Krueger and Y. Bengio 2014 **NICE: Non-linear Independent Components Estimation** e-Print: 1410.8516 doi: 10.48550/arxiv.1410.8516
- [142] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever and M. Welling 2016 **Improved Variational Inference with Inverse Autoregressive Flow** *Proceedings of the 30th International Conference on Neural Information Processing Systems NIPS'16* (Red Hook, NY, USA: Curran Associates Inc.) p 4743 ISBN 9781510838819. e-Print: 1606.04934 doi: 10.48550/arXiv.1606.04934

- 
- [143] G. Papamakarios, T. Pavlakou and I. Murray 2017 **Masked Autoregressive Flow for Density Estimation** *Advances in Neural Information Processing Systems* vol 30 edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (Curran Associates, Inc.). e-Print: 1705.07057 doi: 10.48550/arxiv.1705.07057
- [144] Y. Hao, A. Orlitsky, A. T. Suresh and Y. Wu 2018 **Data Amplification: A Unified and Competitive Approach to Property Estimation** *Advances in Neural Information Processing Systems* vol 31 edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett (Curran Associates, Inc.). e-Print: 1904.00070 doi: 10.48550/arxiv.1904.00070
- [145] H. Dembinski, P. Ongmongkolkul, C. Deil, D. M. Hurtado, M. Feickert, H. Schreiner, Andrew, C. Burr, F. Rost, A. Pearce, L. Geiger, B. M. Wiedemann *et al.* 2020 **scikit-hep/iminuit: v1.4.9** Zenodo doi: 10.5281/zenodo.3951328
- [146] P. Ongmongkolkul, C. Deil, C. hsiang Cheng, A. Pearce, E. Rodrigues, H. Schreiner, M. Marinangeli, L. Geiger and H. Dembinski 2018 **scikit-hep/probfit: 1.1.0** Zenodo doi: 10.5281/zenodo.1477853
- [147] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen and X. Chen 2016 **Improved Techniques for Training GANs** *Advances in Neural Information Processing Systems* vol 29 edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon and R. Garnett (Curran Associates, Inc.). e-Print: 1606.03498 doi: 10.48550/arXiv.1606.03498
- [148] P. T. Komiske, E. M. Metodiev and J. Thaler 2019 **Energy Flow Networks: Deep Sets for Particle Jets** *JHEP* **01** 121. e-Print: 1810.05165 doi: 10.1007/JHEP01(2019)121
- [149] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. R. Salakhutdinov and A. J. Smola 2017 **Deep Sets** *Advances in Neural Information Processing Systems* vol 30 edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (Curran Associates, Inc.). e-Print: 1703.06114 doi: 10.48550/arXiv.1703.06114
- [150] K. Roth, A. Lucchi, S. Nowozin and T. Hofmann 2017 **Stabilizing Training of Generative Adversarial Networks through Regularization** *Proceedings of the 31st International Conference on Neural Information Processing Systems NIPS'17* (Red Hook, NY, USA: Curran Associates Inc.) p 2015 ISBN 9781510860964. e-Print: 1705.09367 doi: 10.48550/arXiv.1705.09367
- [151] A. Paszke *et al.* edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett 2019 **PyTorch: An Imperative Style, High-Performance Deep Learning Library** *Advances in Neural Information Processing Systems* **32** 8024. e-Print: 1912.01703 doi: 10.48550/arXiv.1912.01703
- [152] A. Radford, L. Metz and S. Chintala 2015 **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks** e-Print: 1511.06434 doi: 10.48550/arxiv.1511.06434
- [153] 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



- [154] The iLCSoft Group 2016 **iLCSoft Project Page** [Online; accessed 10-September-2022] URL: <https://github.com/iLCSoft>
- [155] G. R. Khattak, S. Vallecorsa, F. Carminati and G. M. Khan 2022 **Fast simulation of a high granularity calorimeter by generative adversarial networks** *Eur. Phys. J. C* **82** 386. e-Print: 2109.07388 doi: 10.1140/epjc/s10052-022-10258-4
- [156] K. Hara, H. Kataoka and Y. Satoh 2018 **Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?** *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Los Alamitos, CA, USA: IEEE Computer Society) p 6546. e-Print: 1711.09577 doi: 10.1109/CVPR.2018.00685
- [157] S. Otten, S. Caron, W. Swart, M. Beekveld, L. Hendriks, C. Leeuwen, D. Podareanu, R. Austri and R. Verheyen 2021 **Event generation and statistical sampling for physics with deep generative models and a density information buffer** *Nature Communications* **12** 2985. e-Print: 1901.00875 doi: 10.1038/s41467-021-22616-z
- [158] E. Parzen 1962 **On Estimation of a Probability Density Function and Mode** *The Annals of Mathematical Statistics* **33** 1065
- [159] 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 doi: 10.1038/s41592-019-0686-2
- [160] The CALICE collaboration 2011 **Tests of a Particle Flow Algorithm with CALICE test beam data** *Journal of Instrumentation* **6** P07005. e-Print: 1105.3417 doi: 10.1088/1748-0221/6/07/p07005
- [161] D. Heuchel 2022 **Particle Flow Studies with Highly Granular Calorimeter Data** doi: 10.11588/heidok.00031794
- [162] R. Brun, F. Rademakers, P. Canal, A. Naumann, O. Couet, L. Moneta, V. Vassilev, S. Linev, D. Piparo, G. GANIS, B. Bellenot, E. Guiraud *et al.* 2019 **root-project/root: v6.18/02** Zenodo doi: 10.5281/zenodo.3895860
- [163] 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
- [164] 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
- [165] The ATLAS collaboration 2020 **Operation of the ATLAS trigger system in Run 2** *JINST* **15** P10004. e-Print: 2007.12539 doi: 10.1088/1748-0221/15/10/P10004
- [166] The CMS collaboration 2017 **The CMS trigger system** *JINST* **12** P01020. e-Print: 1609.02366 doi: 10.1088/1748-0221/12/01/P01020

- 
- [167] The CMS collaboration 2002 **CMS The TriDAS Project: Technical Design Report, Volume 2: Data Acquisition and High-Level Trigger**. CMS trigger and data-acquisition project Technical design report. CMS (Geneva: CERN) URL: <http://cds.cern.ch/record/578006>
- [168] The ATLAS collaboration 1998 **ATLAS level-1 trigger: Technical Design Report** Technical design report. ATLAS (Geneva: CERN) URL: <https://cds.cern.ch/record/381429>
- [169] The LHCb collaboration 2014 **LHCb Trigger and Online Upgrade Technical Design Report** Tech. rep. CERN Geneva URL: <https://cds.cern.ch/record/1701361>
- [170] The ALICE collaboration 2013 **Upgrade of the ALICE Readout & Trigger System** Tech. rep. CERN Geneva URL: <https://cds.cern.ch/record/1603472>
- [171] J. Duarte *et al.* 2018 **Fast inference of deep neural networks in FPGAs for particle physics** *JINST* **13** P07027. e-Print: 1804.06913 doi: 10.1088/1748-0221/13/07/P07027
- [172] N. Nottbeck, C. Schmitt and V. Büscher 2019 **Implementation of high-performance, sub-microsecond deep neural networks on FPGAs for trigger applications** *JINST* **14** P09014. e-Print: 1903.10201 doi: 10.1088/1748-0221/14/09/p09014
- [173] S. Summers *et al.* 2020 **Fast inference of Boosted Decision Trees in FPGAs for particle physics** *JINST* **15** P05026. e-Print: 2002.02534 doi: 10.1088/1748-0221/15/05/P05026
- [174] The CMS collaboration 2020 **The Phase-2 Upgrade of the CMS Level-1 Trigger** Tech. rep. CERN Geneva final version URL: <https://cds.cern.ch/record/2714892>
- [175] T. M. Hong, B. Carlson, B. Eubanks, S. Racz, S. Roche, J. Stelzer and D. Stump 2021 **Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics** *JINST* **16** P08016. e-Print: 2104.03408 doi: 10.1088/1748-0221/16/08/P08016
- [176] T. Aarrestad *et al.* 2021 **Fast convolutional neural networks on FPGAs with hls4ml** *Mach. Learn. Sci. Tech.* **2** 045015. e-Print: 2101.05108 doi: 10.1088/2632-2153/ac0ea1
- [177] A. M. Deiana *et al.* 2022 **Applications and Techniques for Fast Machine Learning in Science** *Front. Big Data* **5** 787421. e-Print: 2110.13041 doi: 10.3389/fdata.2022.787421
- [178] R. Aaij *et al.* 2016 **Tesla : an application for real-time data analysis in High Energy Physics** *Comput. Phys. Commun.* **208** 35. e-Print: 1604.05596 doi: 10.1016/j.cpc.2016.07.022
- [179] The CMS collaboration 2016 **Search for narrow resonances in dijet final states at  $\sqrt{s} = 8$  TeV with the novel CMS technique of data scouting** *Phys. Rev. Lett.* **117** 031802. e-Print: 1604.08907 doi: 10.1103/PhysRevLett.117.031802

- [180] The ATLAS collaboration 2018 **Search for low-mass dijet resonances using trigger-level jets with the ATLAS detector in  $pp$  collisions at  $\sqrt{s} = 13$  TeV** *Phys. Rev. Lett.* **121** 081801. e-Print: 1804.03496 doi: 10.1103/PhysRevLett.121.081801
- [181] R. Aaij *et al.* 2019 **A comprehensive real-time analysis model at the LHCb experiment** *JINST* **14** P04006. e-Print: 1903.01360 doi: 10.1088/1748-0221/14/04/P04006
- [182] R. M. Karp 1992 **On-Line Algorithms Versus Off-Line Algorithms: How Much is It Worth to Know the Future?** *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I* (NLD: North-Holland Publishing Co.) p 416 ISBN 044489747X
- [183] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov and A. G. Wilson 2019 **Averaging Weights Leads to Wider Optima and Better Generalization** e-Print: 1803.05407 doi: 10.48550/arXiv.1803.05407
- [184] B. Nachman and D. Shih 2020 **Anomaly detection with density estimation** *Phys. Rev. D* **101**(7) 075042. e-Print: 2001.04990 doi: 10.1103/PhysRevD.101.075042
- [185] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann 2020 **Exploring phase space with Neural Importance Sampling** *SciPost Phys.* **8** 069. e-Print: 2001.05478 doi: 10.21468/SciPostPhys.8.4.069
- [186] C. Gao, J. Isaacson and C. Krause 2020 **i-flow: High-dimensional Integration and Sampling with Normalizing Flows** *Mach. Learn. Sci. Tech.* **1** 045023. e-Print: 2001.05486 doi: 10.1088/2632-2153/abab62
- [187] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz 2020 **Event Generation with Normalizing Flows** *Phys. Rev. D* **101** 076002. e-Print: 2001.10028 doi: 10.1103/PhysRevD.101.076002
- [188] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone and U. Köthe 2020 **Invertible Networks or Partons to Detector and Back Again** *SciPost Phys.* **9** 074. e-Print: 2006.06685 doi: 10.21468/SciPostPhys.9.5.074
- [189] B. Stienen and R. Verheyen 2021 **Phase space sampling and inference from weighted events with autoregressive flows** *SciPost Phys.* **10** 038. e-Print: 2011.13445 doi: 10.21468/SciPostPhys.10.2.038
- [190] R. Winterhalder, M. Bellagente and B. Nachman 2021 **Latent Space Refinement for Deep Generative Models** e-Print: 2106.00792
- [191] A. Hallin, J. Isaacson, G. Kasieczka, C. Krause, B. Nachman, T. Quadfasel, M. Schlaffer, D. Shih and M. Sommerhalder 2022 **Classifying anomalies through outer density estimation** *Phys. Rev. D* **106** 055006. e-Print: 2109.00546 doi: 10.1103/PhysRevD.106.055006
- [192] R. Winterhalder, V. Magerya, E. Villa, S. P. Jones, M. Kerner, A. Butter, G. Heinrich and T. Plehn 2022 **Targeting multi-loop integrals with neural networks** *SciPost Phys.* **12** 129. e-Print: 2112.09145 doi: 10.21468/SciPostPhys.12.4.129

- 
- [193] M. Germain, K. Gregor, I. Murray and H. Larochelle 2015 **MADE: Masked Autoencoder for Distribution Estimation** e-Print: 1502.03509
- [194] G. Choudalakis 2011 **On hypothesis testing, trials factor, hypertests and the BumpHunter** *PHYSTAT 2011*. e-Print: 1101.0390 doi: 10.48550/arXiv.1101.0390
- [195] T. Sjostrand, S. Mrenna and P. Z. Skands 2008 **A Brief Introduction to PYTHIA 8.1** *Comput. Phys. Commun.* **178** 852. e-Print: 0710.3820 doi: 10.1016/j.cpc.2008.01.036
- [196] A. Mertens edited by L. Fiala, M. Lokajicek and N. Tumova 2015 **New features in Delphes 3** *J. Phys. Conf. Ser.* **608** 012045 doi: 10.1088/1742-6596/608/1/012045
- [197] 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
- [198] M. Cacciari, G. P. Salam and G. Soyez 2012 **FastJet User Manual** *Eur. Phys. J. C* **72** 1896. e-Print: 1111.6097 doi: 10.1140/epjc/s10052-012-1896-2
- [199] A. Andreassen, B. Nachman and D. Shih 2020 **Simulation Assisted Likelihood-free Anomaly Detection** *Phys. Rev. D* **101** 095004. e-Print: 2001.05001 doi: 10.1103/PhysRevD.101.095004
- [200] J. Thaler and K. Van Tilburg 2012 **Maximizing Boosted Top Identification by Minimizing N-subjettiness** *JHEP* **02** 093. e-Print: 1108.2701 doi: 10.1007/JHEP02(2012)093
- [201] J. Thaler and K. Van Tilburg 2011 **Identifying Boosted Objects with N-subjettiness** *JHEP* **03** 015. e-Print: 1011.2268 doi: 10.1007/JHEP03(2011)015
- [202] E. M. Metodiev, B. Nachman and J. Thaler 2017 **Classification without labels: Learning from mixed samples in high energy physics** *JHEP* **10** 174. e-Print: 1708.02949 doi: 10.1007/JHEP10(2017)174
- [203] J. H. Collins, K. Howe and B. Nachman 2018 **Anomaly Detection for Resonant New Physics with Machine Learning** *Phys. Rev. Lett.* **121** 241803. e-Print: 1805.02664 doi: 10.1103/PhysRevLett.121.241803
- [204] J. H. Collins, K. Howe and B. Nachman 2019 **Extending the search for new resonances with machine learning** *Phys. Rev. D* **99** 014038. e-Print: 1902.02634 doi: 10.1103/PhysRevD.99.014038
- [205] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong and Q. He 2021 **A Comprehensive Survey on Transfer Learning** *Proceedings of the IEEE* **109** 43 doi: 10.1109/JPROC.2020.3004555
- [206] T. Hospedales, A. Antoniou, P. Micaelli and A. Storkey 2022 **Meta-Learning in Neural Networks: A Survey** *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44** 5149. e-Print: 2004.05439 doi: 10.1109/TPAMI.2021.3079209
- [207] D. Ulyanov, A. Vedaldi and V. Lempitsky 2016 **Instance Normalization: The Missing Ingredient for Fast Stylization** e-Print: 1607.08022 doi: 10.48550/arxiv.1607.08022

# Appendix A

## Neural Network Architectures

This appendix gives detailed information about the neural networks used in the photon shower generation (Chapter 6), pion shower generation (Chapter 7), and testbeam data generation (Chapter 8). All models were implemented using PYTORCH [151], and implementations of the photon generation GAN, WGAN, and BIB-AE can be found under [https://github.com/FLC-QU-hep/getting\\_high](https://github.com/FLC-QU-hep/getting_high), while the implementations of the pion generation WGAN and BIB-AE are available at [https://github.com/FLC-QU-hep/neurIPS2021\\_hadron](https://github.com/FLC-QU-hep/neurIPS2021_hadron).

The descriptions of the photon shower models follow Reference [2], and the descriptions of the pion shower models follow Reference [3]

### Photon GAN

The architecture and trained model of the photon GAN were provided by Anatolii Korol. The photon GAN consists of a generator network and a discriminator network. Both models were trained with the ADAM optimizer [108] with a learning rate of  $2 \times 10^{-5}$ . For every generator update step, one discriminator update step is performed. The model was trained for 6 epochs.

The architectures of the photon shower generator and discriminator are shown in Figure A.1.

### Photon WGAN

The architecture and trained model of the photon WGAN were provided by Engin Eren. The photon WGAN consists of a generator network, a critic network, and an energy constrainer network. Both the generator and critic are trained using the ADAM optimizer with an initial learning rate of  $10^{-4}$ . This learning rate is reduced by a factor of 10 after every 50k update steps. For every generator update step, five discriminator update steps are performed. In total, the generator and critic are trained for 20 epochs. The constrainer model is trained in advance, using SGD with a learning rate of  $10^{-5}$ . The constrainer is trained for a total of 30k iterations.

The architectures of the photon shower generator and the critic are shown in Figure A.2. The architecture of the energy constrainer is shown in Figure A.3.

## Photon BIB-AE

The photon BIB-AE model consists of an encoder network, a decoder network, a critic network, a latent critic network, and a Post Processor network. All models are trained using the ADAM optimizer with a learning rate of  $0.5 \times 10^{-3}$ , with the exception of the latent critic, which uses a learning rate of  $2 \times 10^{-3}$ . After every epoch, all learning rates are multiplied by a factor of 0.95. For every generator update step, five discriminator update steps are performed. The BIB-AE is initially trained for 35 epochs without the Post Processor. Following this, the combined BIB-AE and Post Processor models are trained for an additional 4 epochs.

The loss of the encoder and decoder networks comprises several components that were combined by summing over each contribution, weighted by an individual factor  $\beta$ . The factors for the loss components were as follows:

- Critic output:  $\beta = 1$
- Latent critic output:  $\beta = 100$
- Latent KLD:  $\beta = 0.05$
- Latent MMD:  $\beta = 100$

The Post Processor model is initially trained for one epoch using an MSE loss comparing input and output. After this initial epoch, the Sorted Kernel MMD described in Section 6.3 is added. The relative weights between the loss terms are:

- Input-output MSE (Post Processor):  $\beta = 1$
- Sorted Kernel MMD (Post Processor):  $\beta = 5$

The Kernel size of the MMD is  $\alpha = 200$ .

The architectures of the photon shower BIB-AE encoder and decoder are shown in Figure A.4, the photon shower BIB-AE critic is shown in Figure A.5, and the latent critic and Post Processor are shown in Figure A.6.

## Pion WGAN

The architecture and trained model of the pion WGAN were provided by Engin Eren. The pion WGAN consists of a generator network, a critic network, and an energy constrainer network. The generator and critic are trained using the ADAM optimizer with learning rates of  $10^{-4}$  and  $10^{-5}$ , respectively. For every generator update step, five discriminator update steps are performed. In total, the generator and critic are trained for 82 epochs. The constrainer model is trained in advance, using ADAM with a learning rate of  $10^{-4}$ .

The architectures of the pion shower generator and energy constrainer are shown in Figure A.7. The architecture of the critic is shown in Figure A.8. The Instancenorm layers used in the critic are described in more detail in Reference [207].

## Pion BIB-AE

The pion BIB-AE model consists of an encoder network, a decoder network, a critic network, a latent critic network, and a Post Processor network. All models are trained using the ADAM optimizer with a learning rate of  $0.5 \times 10^{-4}$ , with the exception of the latent critic, which uses a learning rate of  $2 \times 10^{-4}$ . After every epoch, all learning rates are multiplied by a factor of 0.97. For every generator update step, five discriminator update steps are performed. The BIB-AE is initially trained for 37 epochs without the Post Processor. Following this, BIB-AE weights are frozen and the Post Processor model is trained for an additional 105 epochs.

The loss of the encoder and decoder networks comprises several components that were combined by summing over each contribution, weighted by an individual factor  $\beta$ . The factors for the loss components were as follows:

- Critic output:  $\beta = 1$
- Latent critic output:  $\beta = 100$
- Latent KLD:  $\beta = 0.01$
- Latent MMD:  $\beta = 100$

The Post Processor model is initially trained for one epoch using an MSE loss comparing input and output. After this initial epoch, the Post Processor loss terms described in Section 7.3 are added. The relative weights between the loss terms are:

- Input-output MSE (Post Processor):  $\beta = 1$
- Sorted Kernel MMD 1 (Post Processor):  $\beta = 5$
- Sorted Kernel MMD 2 (Post Processor):  $\beta = 5$
- Sorted MSE (Post Processor):  $\beta = 10$
- Sorted MAE (Post Processor):  $\beta = 10$
- Batch comparison (Post Processor):  $\beta = 0.0001$

The Kernel sizes are  $\alpha_1 = 40$  and  $\alpha_2 = 4$  for MMD 1 and MMD 2 respectively.

The architectures of the pion shower BIB-AE encoder and decoder are shown in Figure A.9, and the pion shower BIB-AE critic is shown in Figure A.10. The critic makes use of a mini-batch discrimination block which is separately shown in Figure A.11. The latent critic and Post Processor are shown in Figure A.12.

## Testbeam BIB-AE

Compared to the pion and photon BIB-AEs, the testbeam data BIB-AE model is additionally conditioned on the targeted visible energy sum in units of GeV. The testbeam BIB-AE consists of an encoder network, a decoder network, a critic network, a latent critic network, and a Post Processor network. All models are trained using the ADAM optimizer with a learning rate of

$0.5 \times 10^{-4}$ , with the exception of the latent critic, which uses a learning rate of  $2 \times 10^{-4}$ . After every epoch, all learning rates are multiplied by a factor of 0.97. For every generator update step, five discriminator update steps are performed. The BIB-AE is initially trained for 82 epochs without the Post Processor. Following this, BIB-AE weights are frozen and the Post Processor model is trained for an additional 19 epochs.

The loss of the encoder and decoder networks comprises several components that were combined by summing over each contribution, weighted by an individual factor  $\beta$ . The factors for the loss components were as follows:

- Critic output:  $\beta = 1$
- Latent critic output:  $\beta = 100$
- Latent KLD:  $\beta = 0.01$
- Latent MMD:  $\beta = 100$

The Post Processor model is initially trained for one epoch using an MSE loss comparing input and output. After this initial epoch, the Post Processor loss terms described in Section 7.3 are added. The relative weights between the loss terms are:

- Input-output MSE (Post Processor):  $\beta = 1$
- Sorted Kernel MMD 1 (Post Processor):  $\beta = 5$
- Sorted Kernel MMD 2 (Post Processor):  $\beta = 5$
- Sorted MSE (Post Processor):  $\beta = 10$
- Sorted MAE (Post Processor):  $\beta = 10$
- Batch comparison (Post Processor):  $\beta = 0.0001$

The Kernel sizes are  $\alpha_1 = 1$  and  $\alpha_2 = 0.1$  for MMD 1 and MMD 2 respectively.

The architectures of the testbeam data BIB-AE encoder and decoder are shown in Figure A.13, the testbeam data BIB-AE critic is shown in Figure A.14. The critic makes use of a minibatch discrimination block which is identical to the one used in the pion BIB-AE critic. This identical minibatch discrimination block is shown in Figure A.11. The latent critic and Post Processor are shown in Figure A.15.



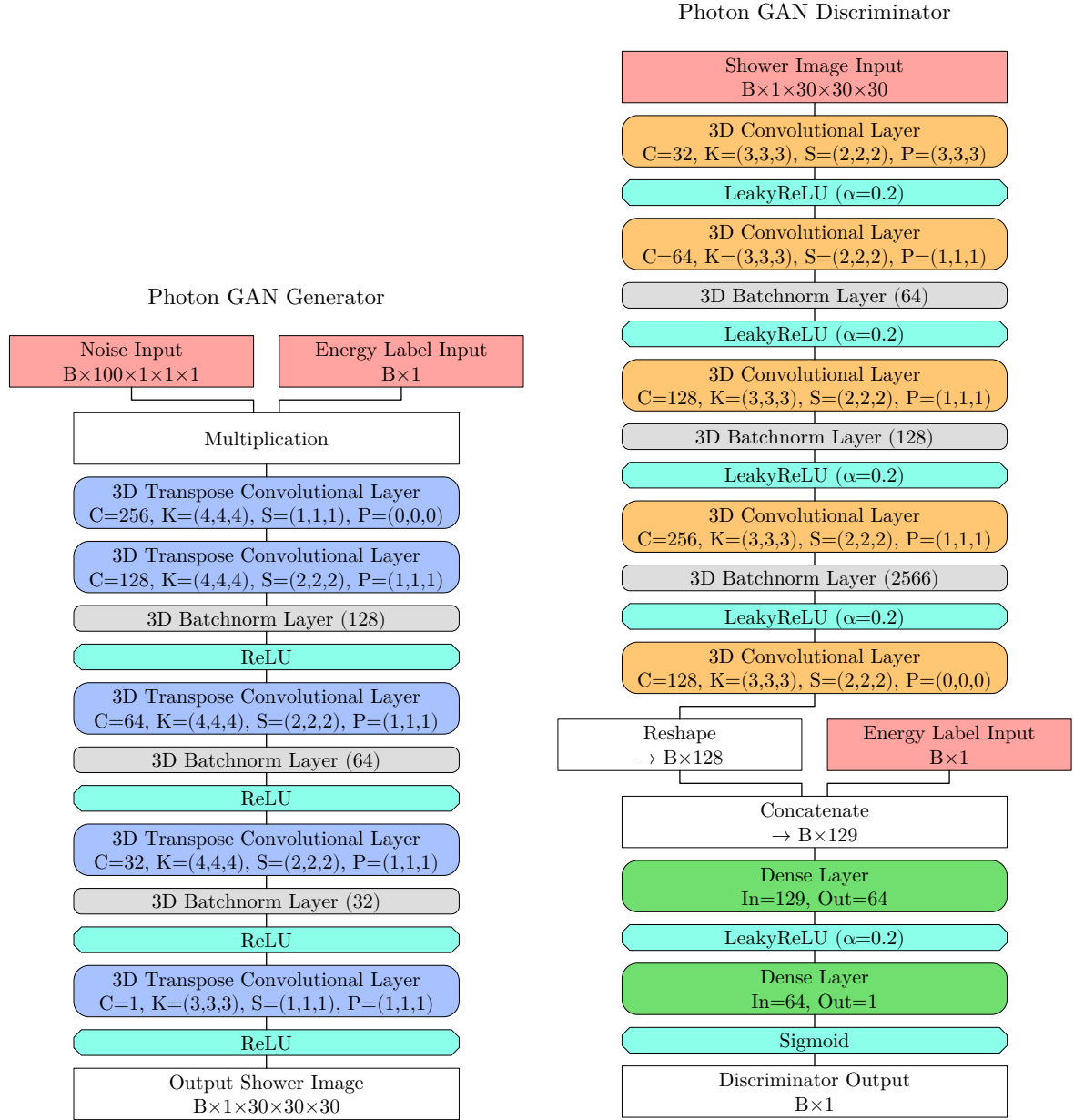


Figure A.1: Architectures of the generator (left) and discriminator (right) of the photon shower GAN. The abbreviation  $B$  describes the batch size,  $C$  is the number of output channels,  $K$  is the kernel size,  $S$  is the stride, and  $P$  is the padding size of a convolution. The numbers after the Batchnorm Layer indicate the shape over which the normalization occurs.

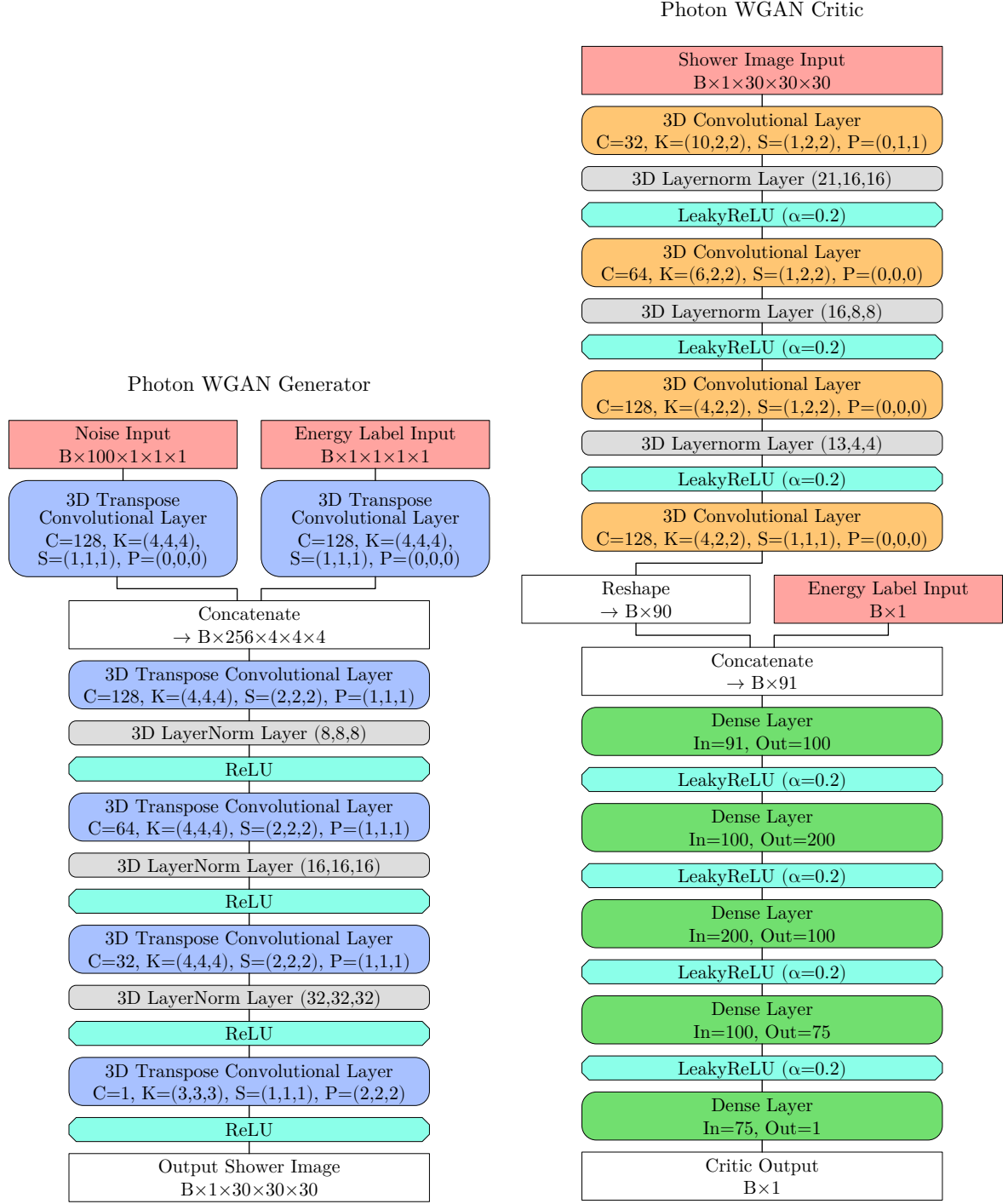


Figure A.2: Architectures of the generator (left) and critic (right) of the photon shower WGAN. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

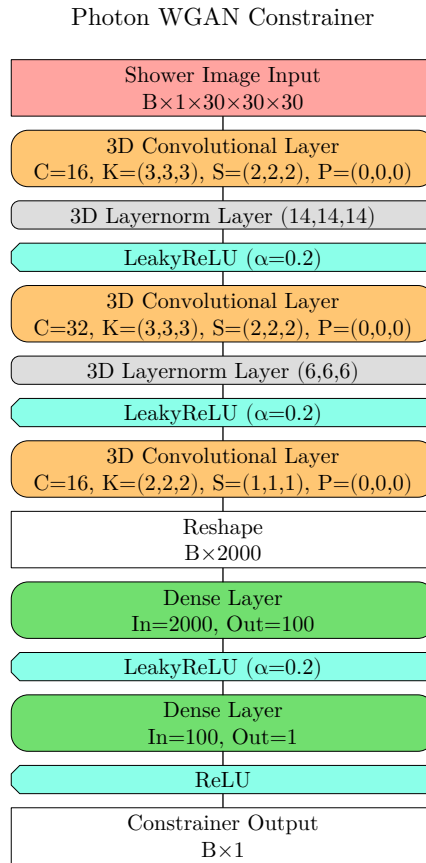


Figure A.3: Architecture photon shower WGAN energy constrainer. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the Layernorm Layer indicate the shape over which the normalization occurs.

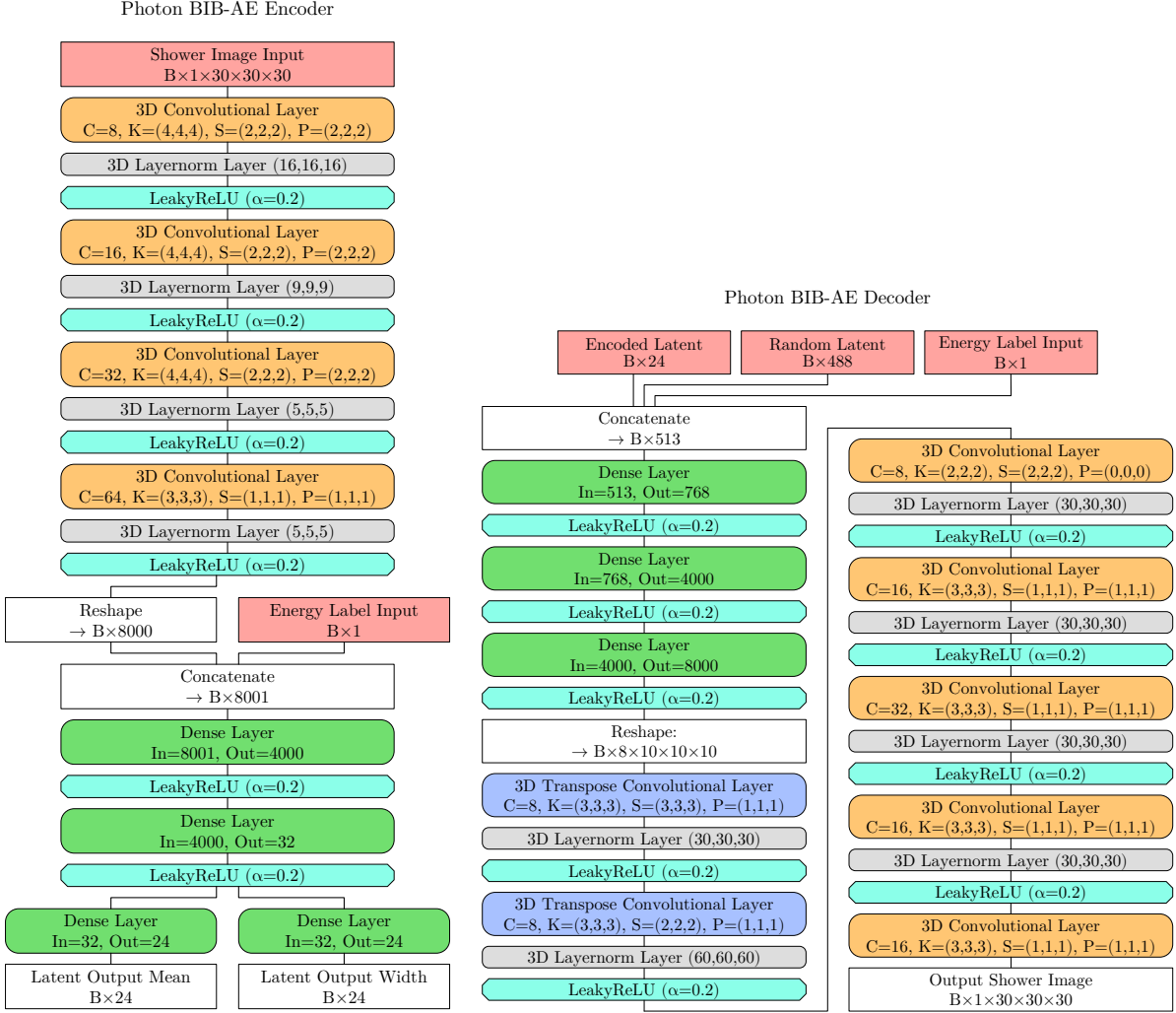


Figure A.4: Architectures of the encoder (left) and decoder (right) of the photon shower BIB-AE. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

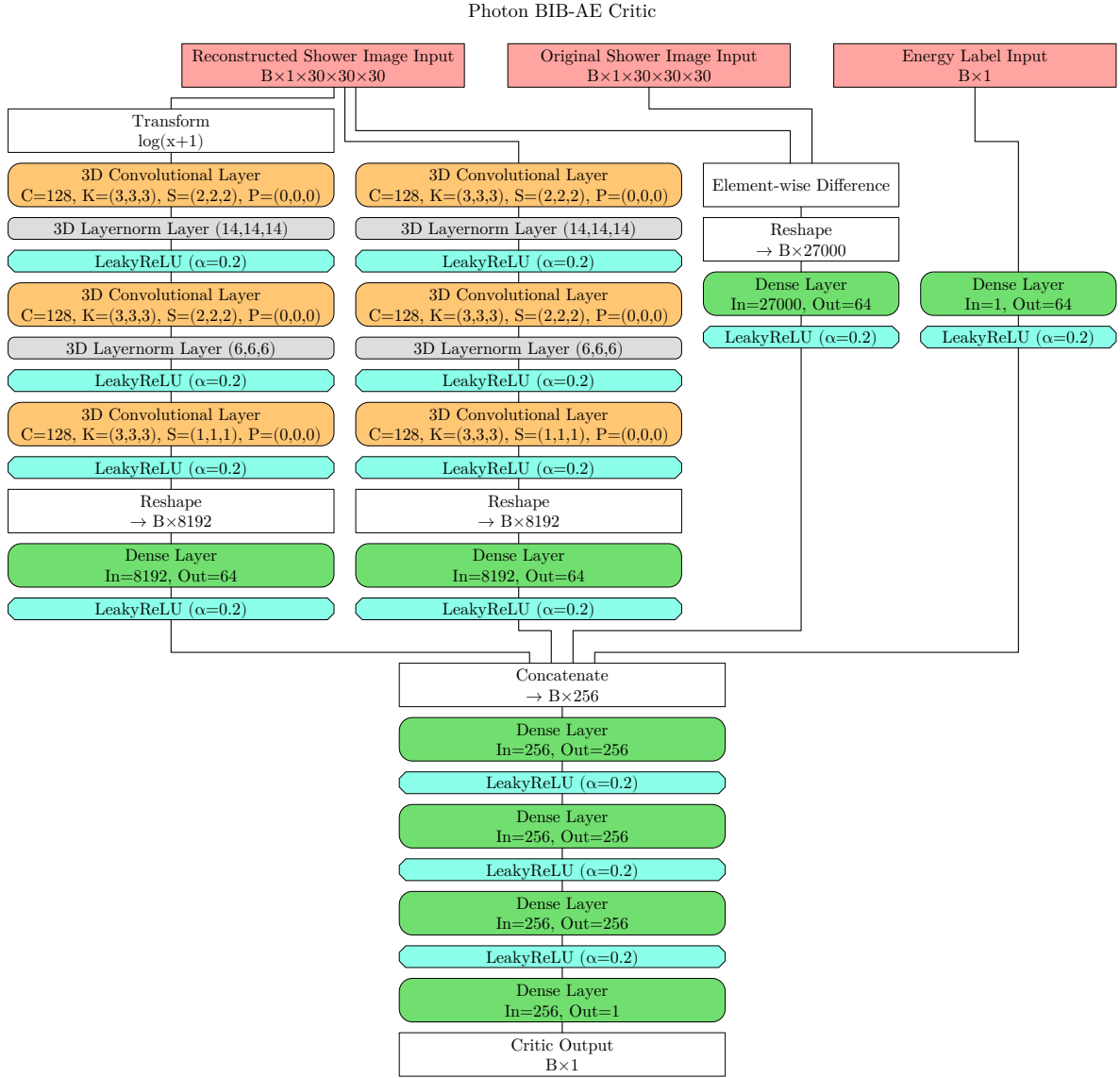


Figure A.5: Architecture of the photon shower BIB-AE critic. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the Layernorm Layer indicate the shape over which the normalization occurs.

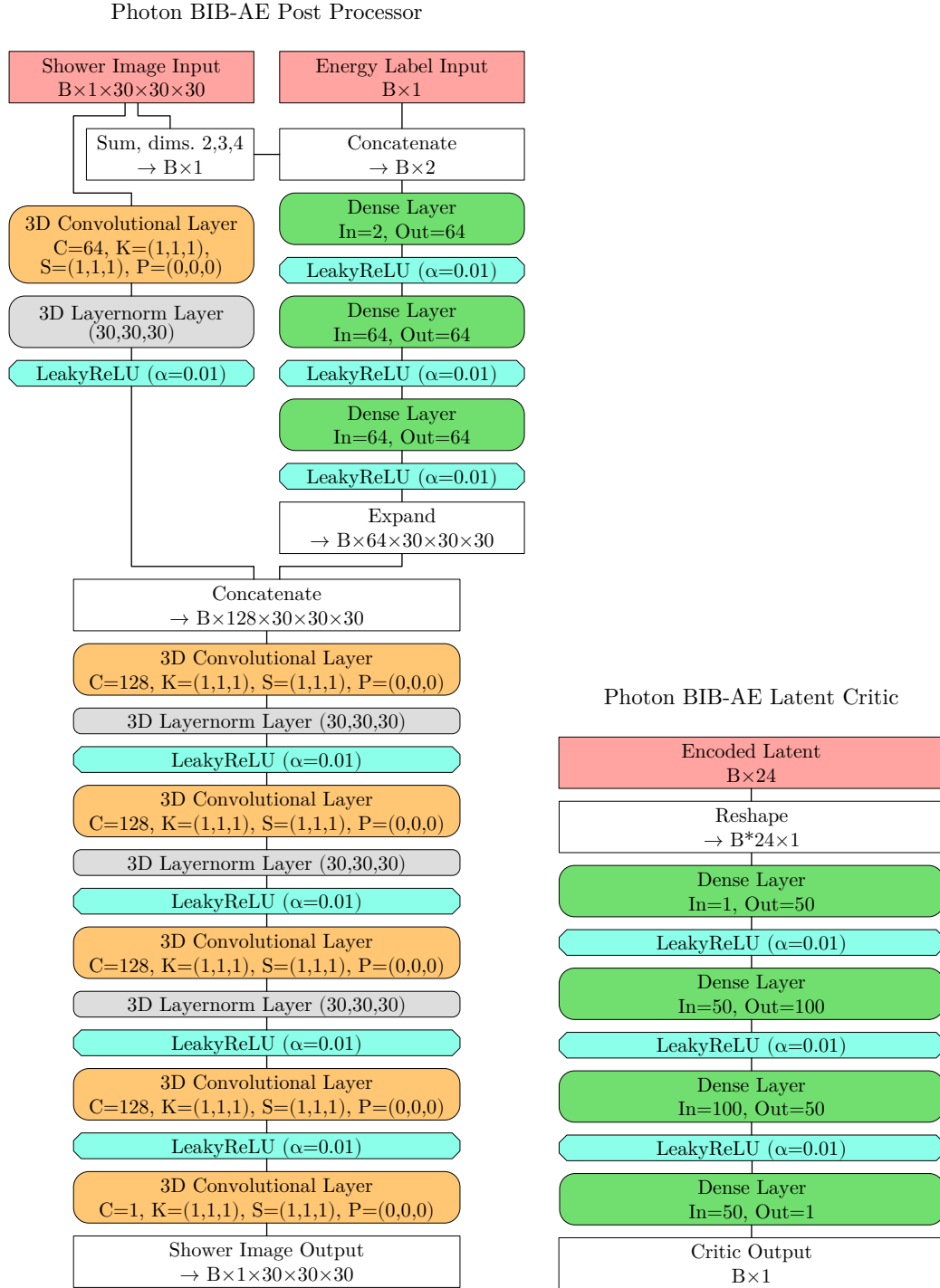


Figure A.6: Architectures of the Post Processor (left) and latent critic (right) of the photon shower BIB-AE. The abbreviation  $B$  describes the batch size,  $C$  is the number of output channels,  $K$  is the kernel size,  $S$  is the stride, and  $P$  is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

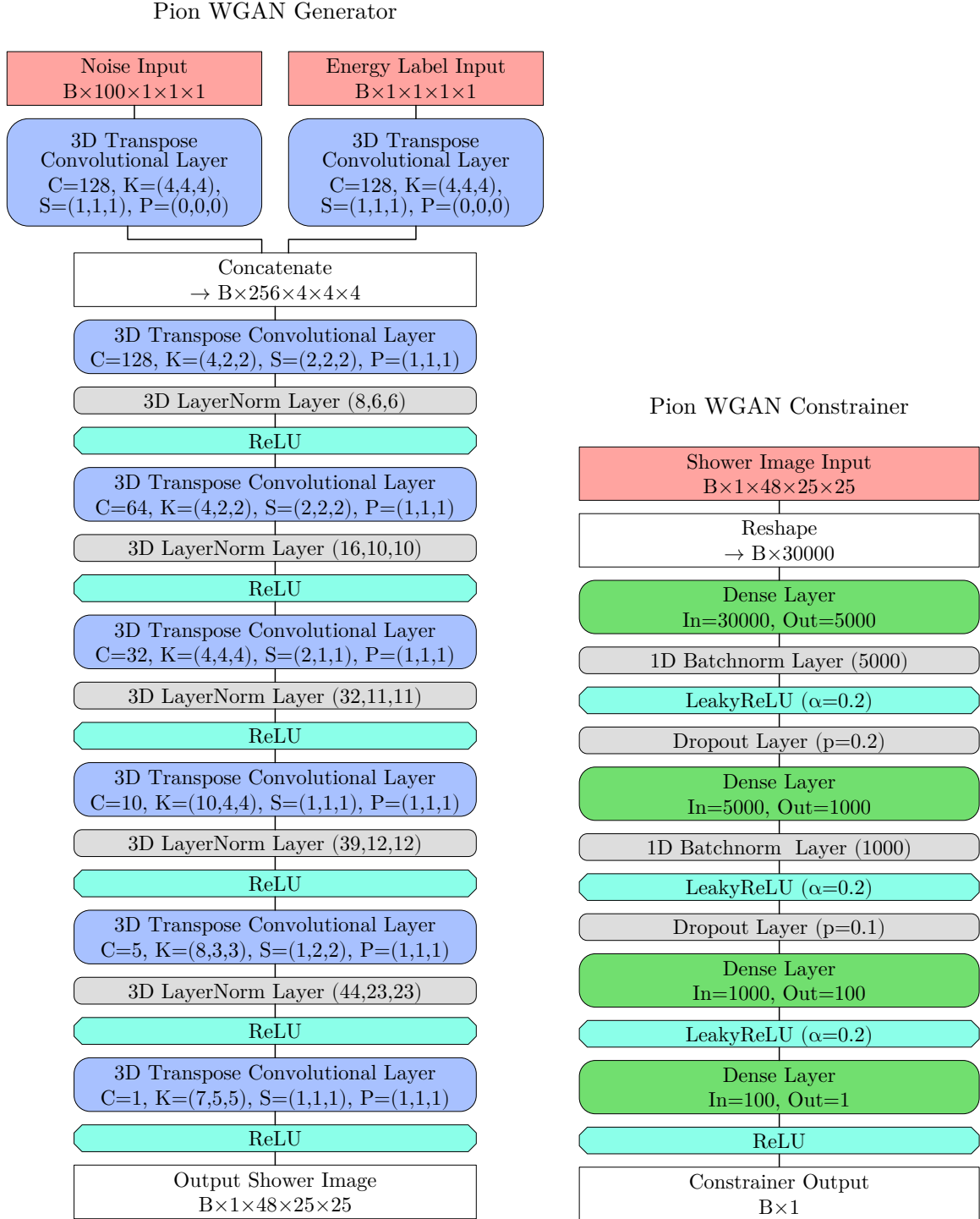


Figure A.7: Architectures of the generator (left) and energy constrainer (right) of the pion shower WGAN. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm and Batchnorm Layer indicate the shape over which the normalization occurs.

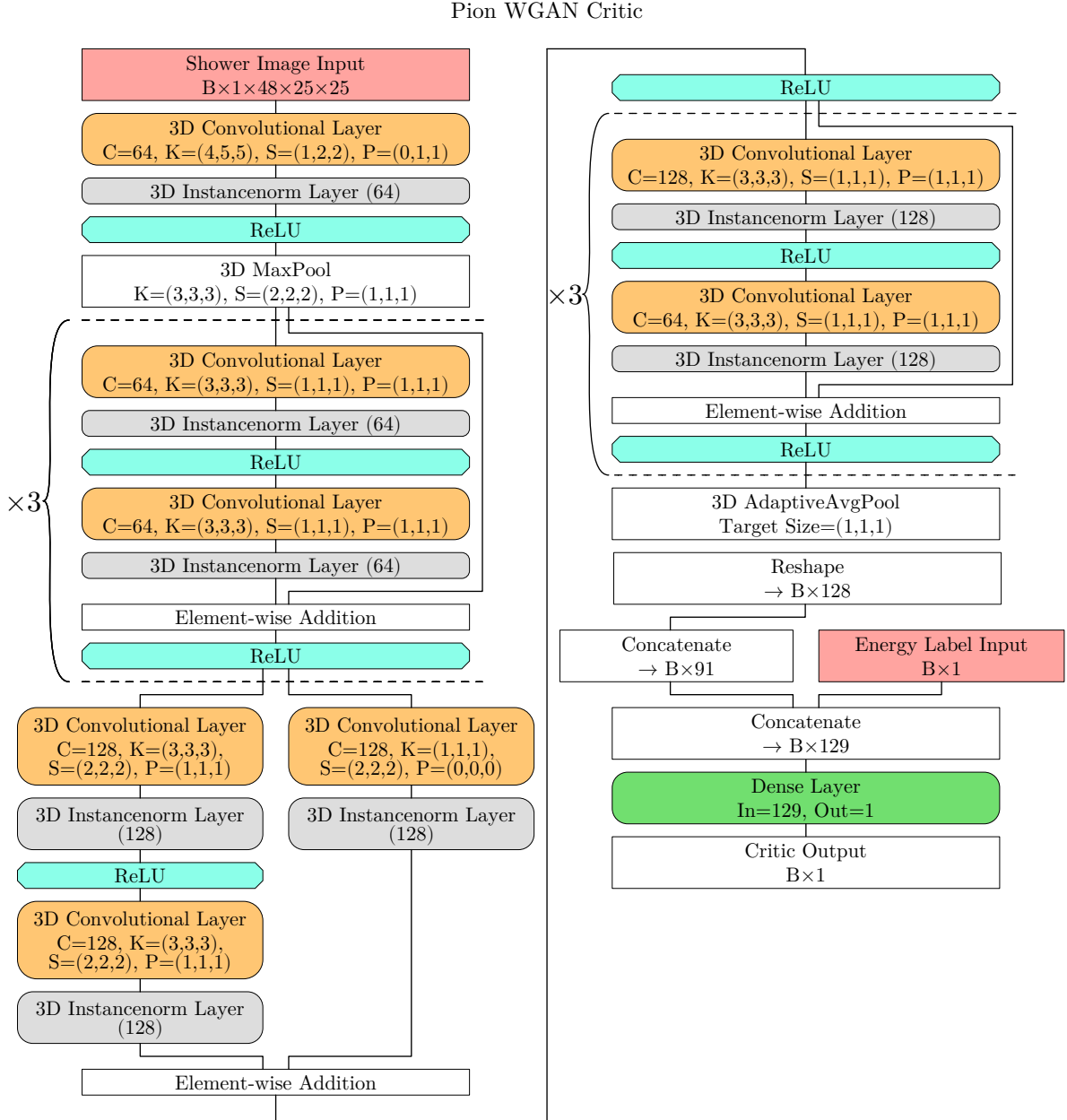


Figure A.8: Architecture of the pion shower WGAN critic. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the Instancenorm Layer indicate the shape over which the normalization occurs.



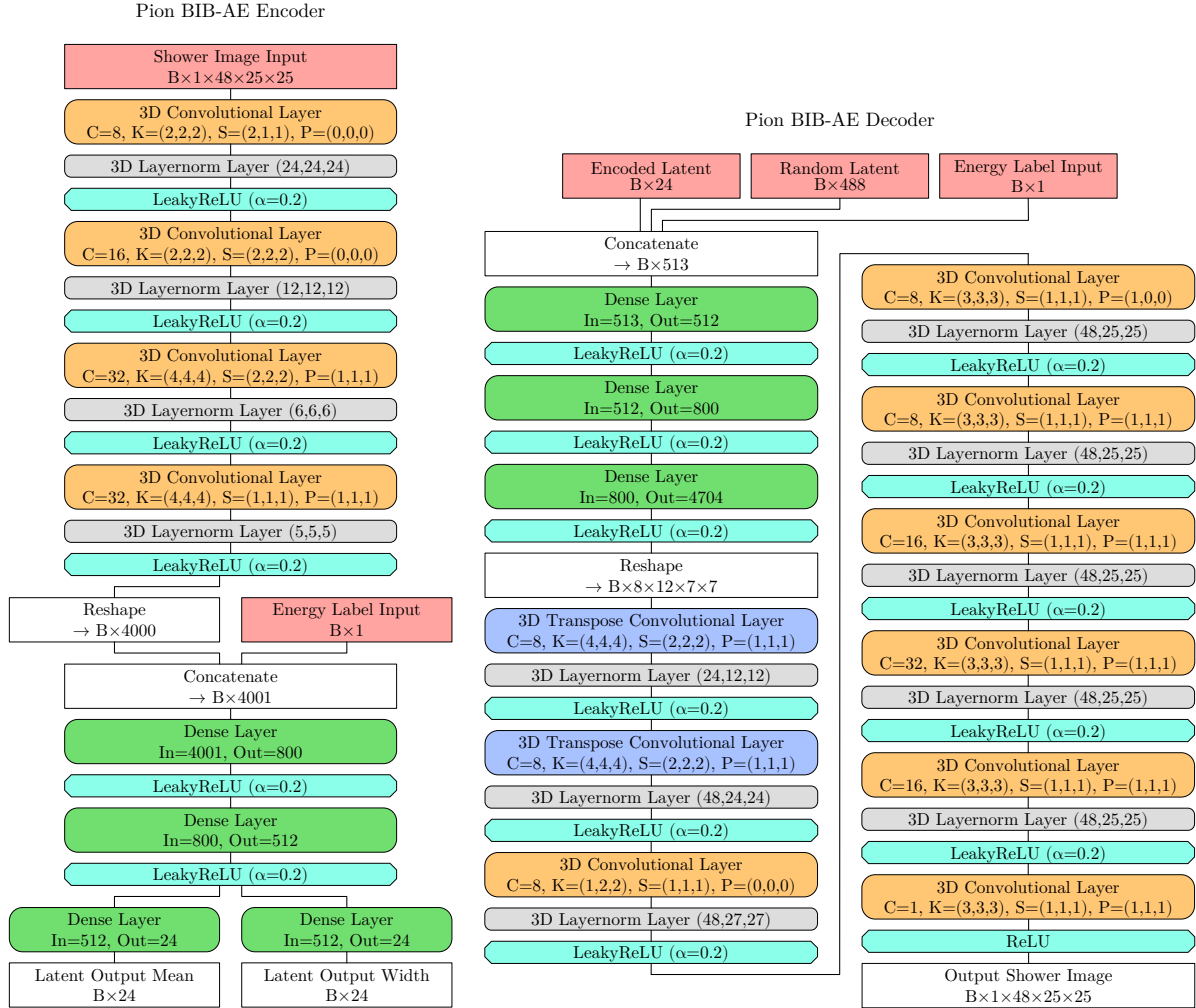


Figure A.9: Architectures of the encoder (left) and decoder (right) of the pion shower BIB-AE. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

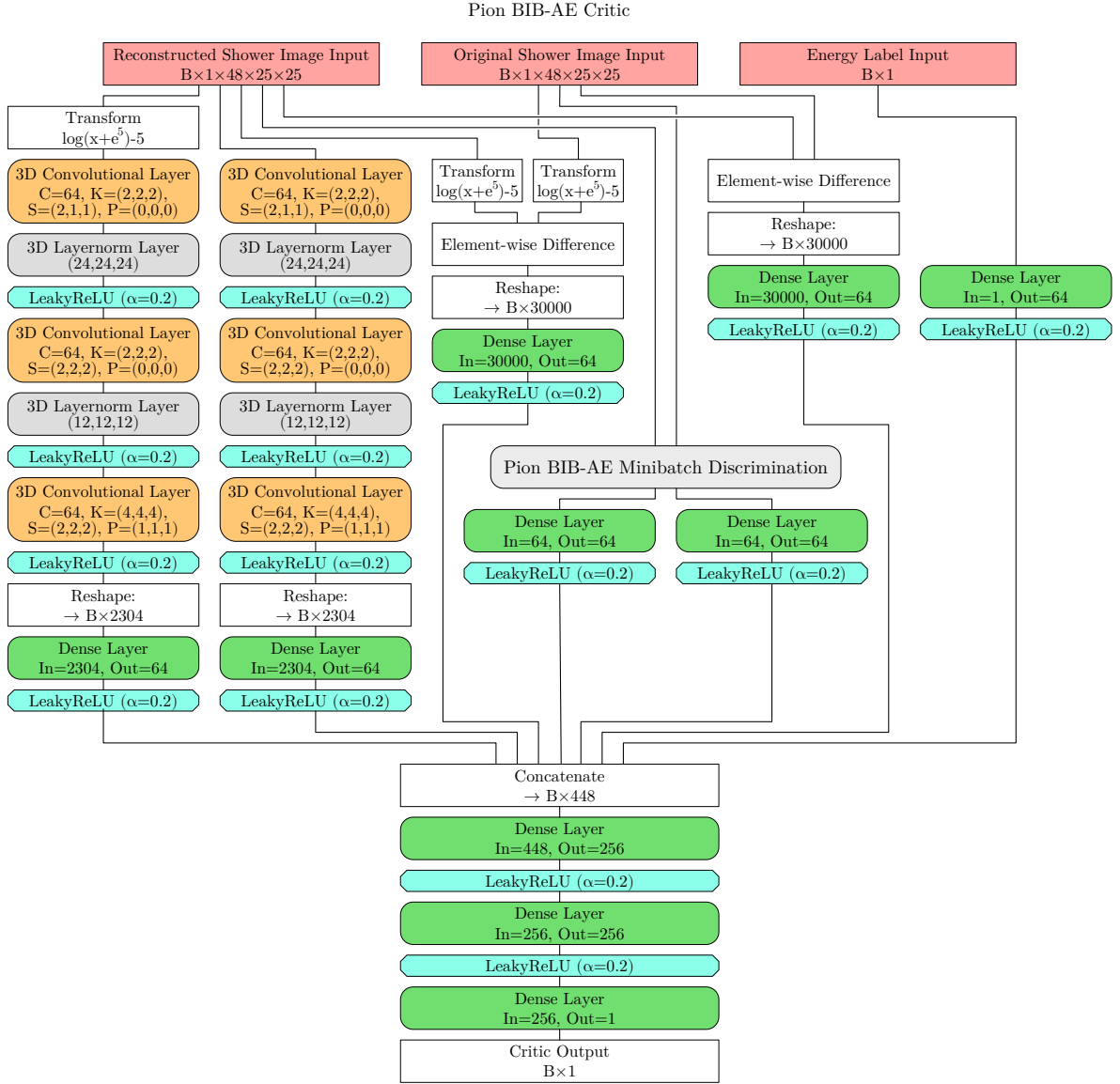


Figure A.10: Architecture of the pion shower BIB-AE critic. The minibatch discrimination block is shown in detail in Figure A.11. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

Pion BIB-AE Minibatch Discrimination

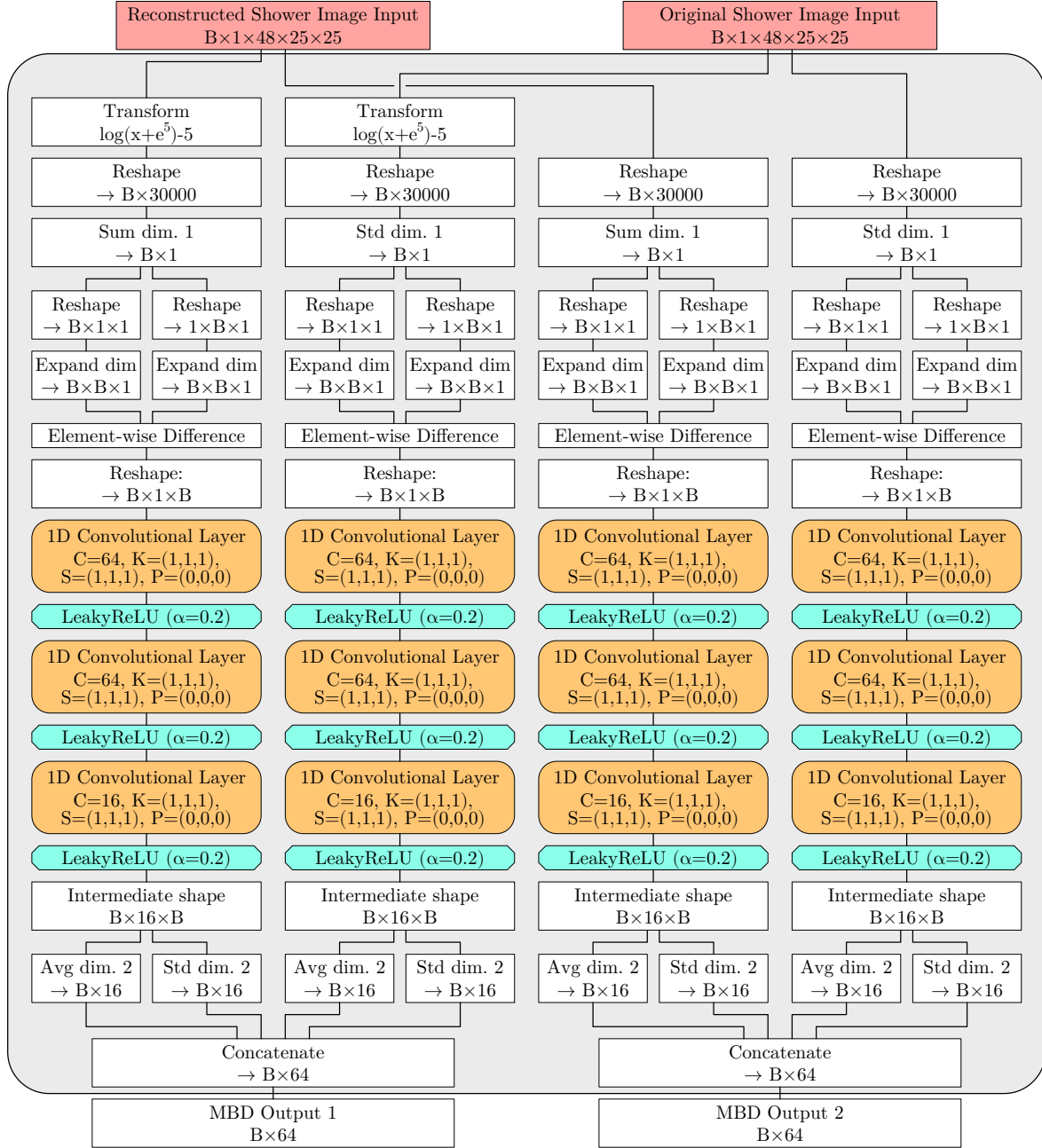


Figure A.11: Architecture of the pion shower BIB-AE minibatch discrimination block used in the BIB-AE critic. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

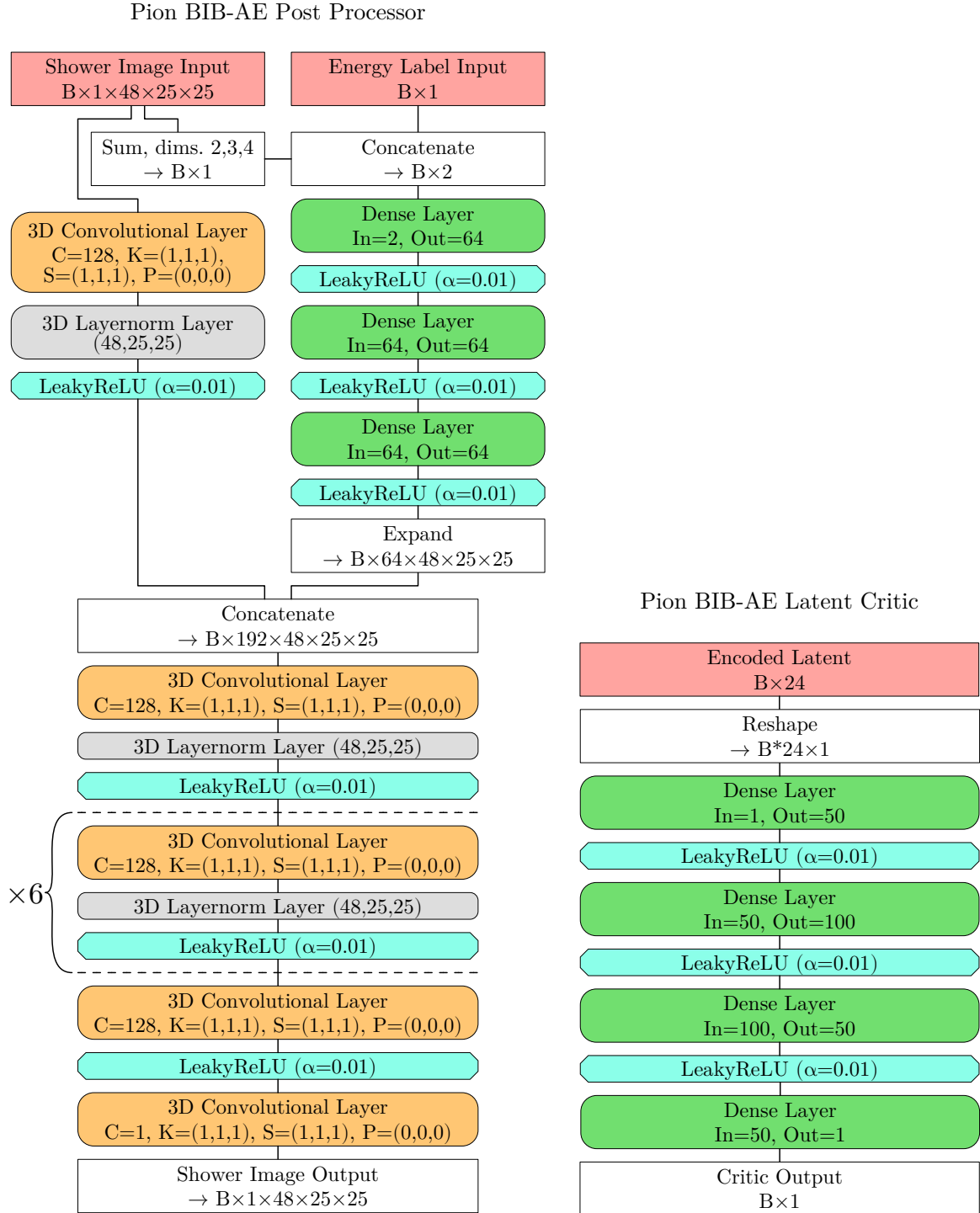


Figure A.12: Architectures of the Post Processor (left) and latent critic (right) of the pion shower BIB-AE. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

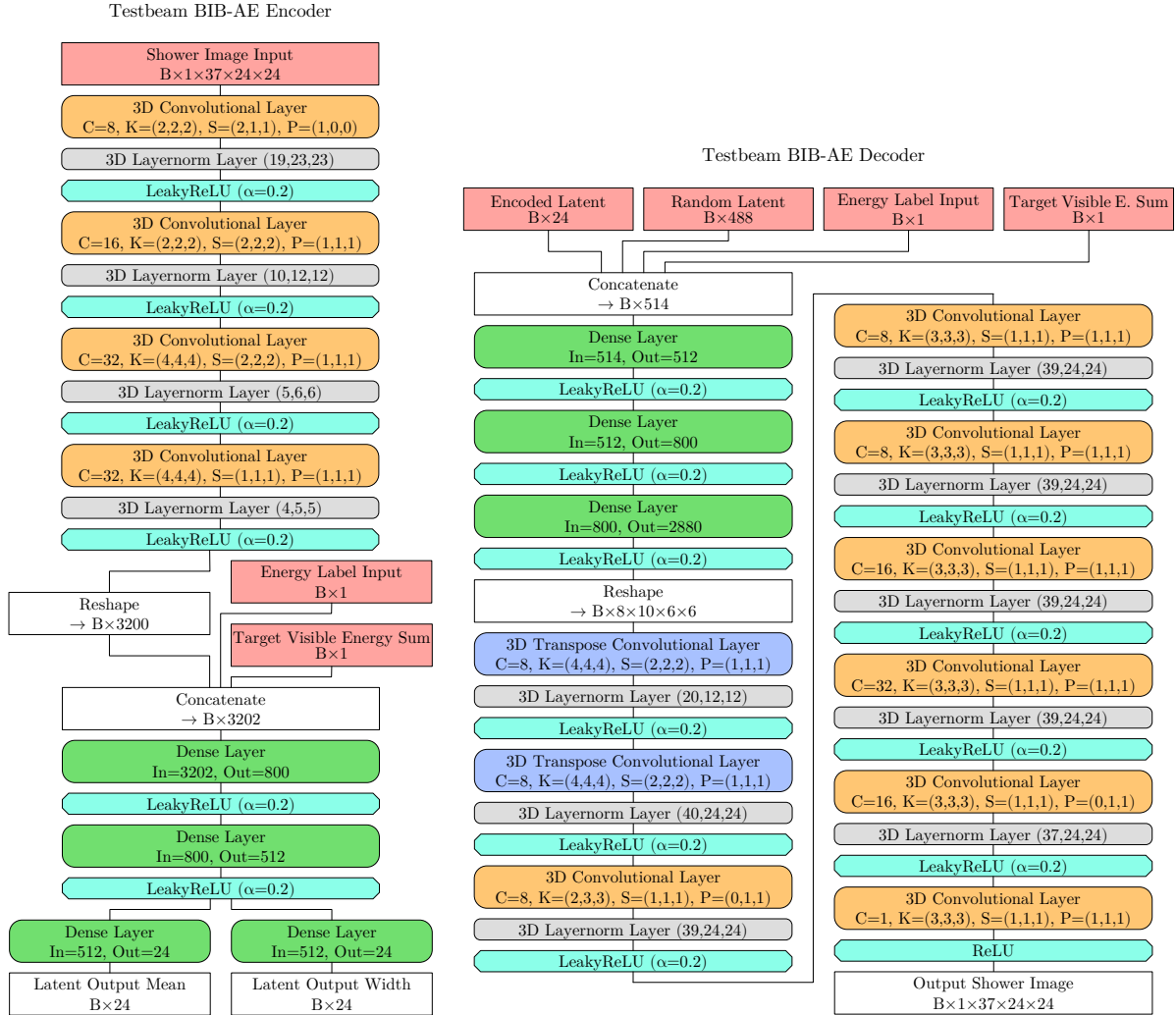


Figure A.13: Architectures of the encoder (left) and decoder (right) of the testbeam data BIB-AE. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

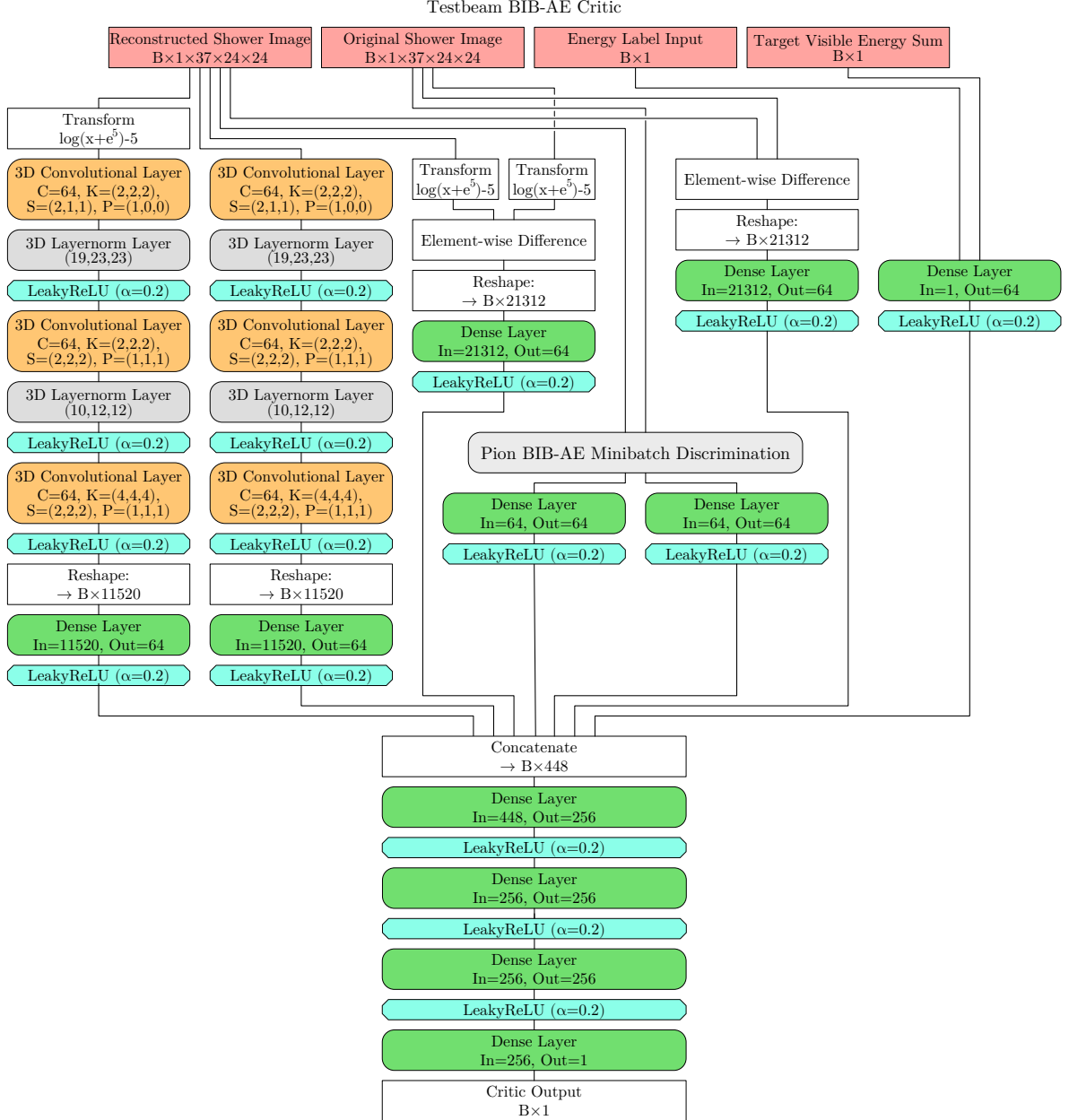


Figure A.14: Architecture of the testbeam data BIB-AE critic. The Minibatch discrimination block is shown is identical to the one used in the pion BIB-AE and is shown in detail in Figure A.11. The abbreviation B describes the batch size, C is the number of output channels, K is the kernel size, S is the stride, and P is the padding size of a convolution. The numbers after the LayerNorm Layer indicate the shape over which the normalization occurs.

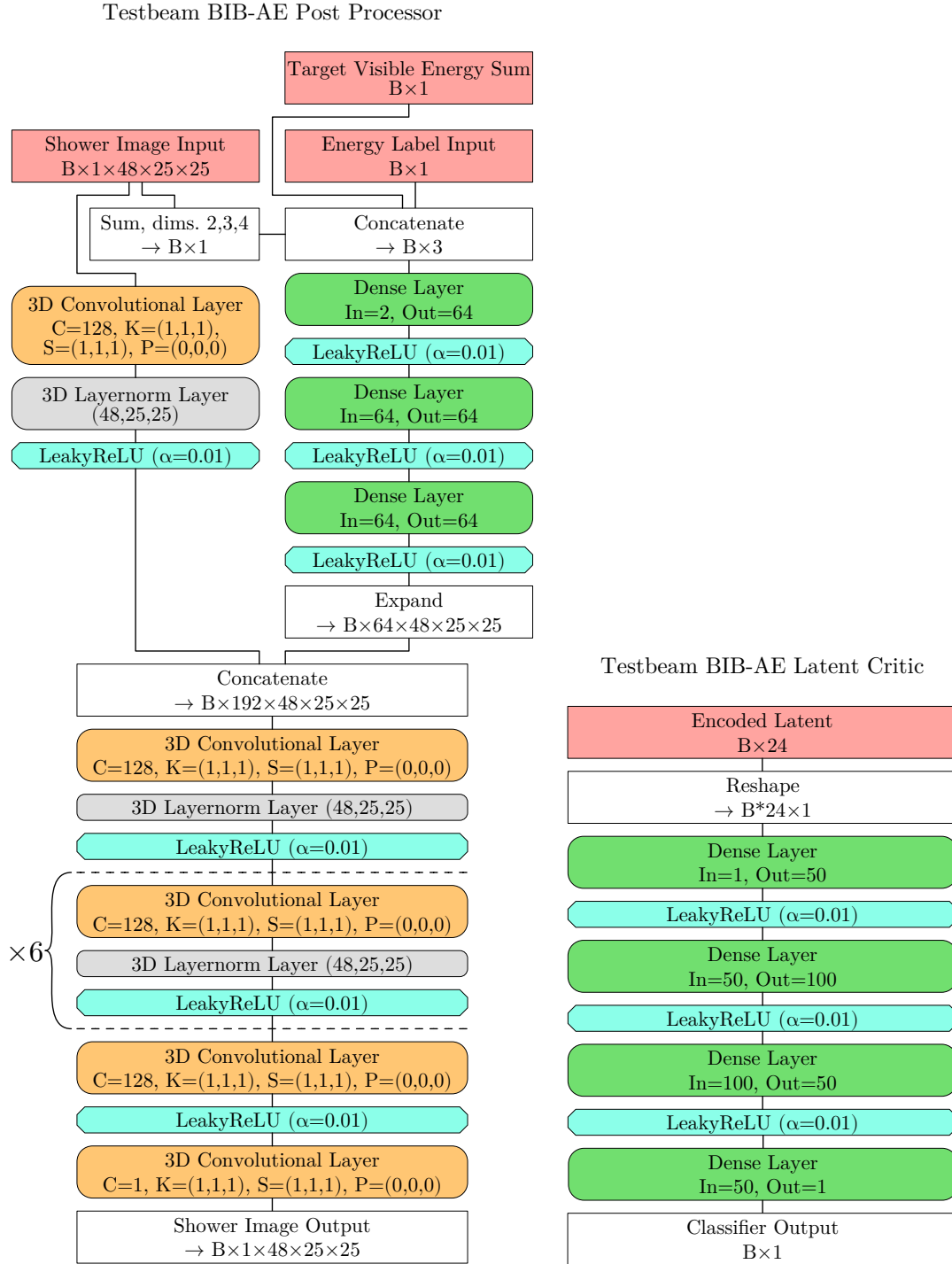


Figure A.15: Architectures of the Post Processor (left) and latent critic (right) of the testbeam data BIB-AE. The abbreviation  $B$  describes the batch size,  $C$  is the number of output channels,  $K$  is the kernel size,  $S$  is the stride, and  $P$  is the padding size of a convolution. The numbers after the Layernorm Layer indicate the shape over which the normalization occurs.





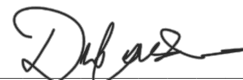
# Eidesstattliche Versicherung

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

Hamburg, den 18.01.2023

---

Ort, Datum



---

Sascha Daniel Diefenbacher