Speeding up Discoveries in the Era of Machine Learning

Accurate and precise density estimation for the LHC

Dissertation

Luigi Favaro

Dissertation

submitted to the

Combined Faculty of Mathematics, Engineering and Natural Sciences
of Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

Luigi Favaro

born in Castrovillari, Italy

Oral examination: 24.10.2024

Speeding up Discoveries in the Era of Machine Learning

Accurate and precise density estimation for the LHC

Referees: Prof. Dr. Tilman Plehn
Prof. Dr. Fred Hamprecht

Abstract

A new era of measurements and tests of the Standard Model of particle physics at the Large Hadron Collider has been shaped by novel methodologies applied to simulations and data analysis. Machine learning is leading this revolution with constant and progressive development of new tools for understanding our data. We have access to techniques that exploit correlations in high-dimensional phase spaces in a statistically principled way. In the context of colliders, these techniques not only boost searches for physics beyond the Standard Model but also improve the already advanced simulation chain. We consider two obstacles that will be critical for the LHC. We propose fast, accurate, and precise surrogate models for simulating the detector response, answering the increasing demand for simulations in the future runs of the LHC. Second, we develop tools for searches of new physics that do not rely on assumptions of the specific new physics signature These tools aim to complement the current paradigm of direct tests of extensions of the SM, which can carry limiting assumptions. We unify these two applications under the lens of precise density estimation using modern machine learning tools, and we demonstrate the importance of using powerful representations that leverage our physics knowledge, e.g. symmetries.

Zusammenfassung

Die Einführung neuer Methoden und Techniken für Simulationen und Datenanalyse hat eine neue Ära von Tests und Messungen des Standardmodels am Large Hadron Collider geprägt. Diese Revolution wird dabei vor allem durch das Machine Learning und dessen ständigen Fortschritt bei der Entwicklung neuer Werkzeuge zum Verständnis unserer Daten angetrieben. Diese erlauben es uns nun Korrelationen im hochdimensionalen Phasenraum auf statistisch fundierte Weise auszunutzen. Im Kontext von Teilchenbeschleunigern hilft dies nicht nur bei der Suche nach Physik jenseits des Standardmodels, sondern verbessert auch die so schon weit fortgeschrittene Simulationskette. Wir betrachten hier zwei kritische Hürden für den LHC. Zum einen schlagen wir schnelle, präzise und akkurate, sogenannte Surrogate Models für die Simulation von Detektoreffekten vor, um dem steigenden Bedarf an Simulationen am LHC nachzukommen. Zum anderen entwickeln wir Werkzeuge für die Suche nach neuer Physik, welche unabhängig von spezifischen Annahmen über dessen Signatur ist. Diese sollen komplementär zum momentanen Paradigma direkter Suchen nach neuer Physik sein, welche auf einschränkenden Annahmen beruhen können. Beide diese Anwendungen vereinen wir unter der Nutzung von Precise Density Estimation mithilfe moderner Machine Learning tools. Hierbei demonstrieren wir die Stärke von Repräsentationen durch die wir unser physikalisches Wissen, wie z.B. Symmetrien, effektiv nutzen können.

Contents

List of Abbreviations

AE	Autoencoder
AUC	Area Under the Curve
BNS	Bespoke Non-stationary Solver
BSM	Beyond Standard Model
CD	Contrastive Divergence
CFM	Conditional Flow Matching
CKM	Cabibbo-Kobayashi-Maskawa
CLR	Contrastive Learning
DGLAP	Dokshitzer-Gribov-Lipatov-Altarelli-Parisi
EBM	Energy-Based Model
ECal	Electromagnetic calorimeter
ELBO	Evidence Lower Bound
FSR	Final State Radiation
GD	Gradient Descent
HCal	Hadronic calorimeter
HEP	High-Energy Physics
HL-LHC	High Luminosity-LHC
IAF	Inverse Autoregressive Flow
INN	invertible neural network
ISR	Initial State Radiation
KL	Kullback-Leibler
LCT	Linear Classifier Test

LHC	Large Hadron Collider
LMC	Langevin Markov Chain
MAF	Masked Autoregressive Flow
MCMC	Markov Chain Monte Carlo
MET	Missing Transverse Energy
ML	Machine Learning
MMD	Maximum Mean Discrepancy
MSE	Mean Squared Error
NAE	Normalized Autoencoder
NF	Normalizing Flow
NLL	Negative Log-Likelihood
NN	Neural Networks
NP	Neyman-Pearson
ODE	Ordinary Differential Equation
OMI	On-Manifold Initialization
OOD	Out-Of-Ditribution
PDF	Parton Distribution Function
QCD	Quantum Chromodynamics
QED	Qunatum Electrodynamics
QFT	Quantum Field Theory
RK	Runge-Kutta
ROC	Receiver Operating Characteristic
SI	Significance Improvement
SM	Standard Model
VAE	Variational Autoencoder
ViT	Vision Transformer

Preface

The research presented in this thesis was conducted at the Institute for Theoretical Physics at Heidelberg University from November 2021 to May 2024. The contents of Chapters 4 and 5 are based on work done in collaboration with other researchers and have been previously published as

[1] Barry M. Dillon, Luigi Favaro, Tilman Plehn, Peter Sorrenson, Michael Krämer,
A normalized autoencoder for LHC triggers,
SciPost Phys. Core 6, 074 (2023), arXiv:2206.14225 [hep-ph]

[2] Ranit Das, Luigi Favaro, Theo Heimel, Claudius Krause,
Tilman Plehn, David Shih,
How to Understand Limitations of Generative Networks,
SciPost Phys. 16 (2024) 031, arXiv:2305.16774 [hep-ph];

[3] Barry M. Dillon, Luigi Favaro, Friedrich Feiden, Tanmoy Modak, Tilman Plehn
Anomalies, representations, and self-supervision,
Submitted to Scipost Phys., arXiv:2301.04660 [hep-ph];

[4] Luigi Favaro, Michael Krämer, Tanmoy Modak, Tilman Plehn, Jan Rüschkamp,
Semi-visible jets, energy-based models, and self-supervision,
Submitted to Scipost Phys., arXiv:2312.03067 [hep-ph];

[5] Florian Ernst, Luigi Favaro, Claudius Krause, Tilman Plehn, David Shih,
Normalizing flows for high-dimensional detector simulations,
Submitted to Scipost Phys., arXiv:2312.09290 [hep-ph];

[6] Luigi Favaro, Ayodele Ore, Sofia Palacios Schweitzer, Tilman Plehn,
CaloDREAM – Detector Response Emulation via Attentive flow Matching,
Submitted to Scipost Phys., arXiv:2405.09629 [hep-ph].

The author is also involved in ongoing projects that have not been ready for publication at the time of writing this thesis.

CHAPTER **1**

Introduction

Our understanding of Nature has advanced considerably since we started studying the subatomic world. We quickly understood, although not without heated discussions, that subatomic particles behave differently than the classical mechanics we more commonly encounter everyday. These observations ultimately gave us quantum field theory (QFT), a theoretical framework for describing free particles and their interactions. Thanks to QFT, we have the Standard Model of particle physics (SM). A predictive model that describes our current understanding of the universe. However, the SM is not perfect and it does not describe completely what we observe nowadays. As physicists, we aim to always strive for a better model of Nature and, secretly, to bring down the SM. The main place where we look for tests of the SM is at particle colliders. Here, collisions between particles accelerated to a fraction of the speed of light giving access to studies of the interactions with large amounts of available energy. The leading experiment in particle physics is the Large Hadron Collider (LHC). Built by the European Organization for Nuclear Research, it is a 27 km long circular collider that accelerates protons to an energy of about 7 GeV.

In recent years the machine learning (ML) revolution has been changing society in every field, including fundamental sciences. In high-energy physics (HEP), ML is allowing for a more systematic approach to physics questions. The ability to model high-dimensional spaces and all the correlations between observables gives ML an edge over previous methods in several tasks, e.g. optimal observables, fast simulators, reduced assumptions, BSM searches, and uncertainty estimation. All of this by leveraging the powerful simulators we can access in particle physics. Indeed simulations are a defining aspect of LHC physics, bridging experiment and fundamental theory and allowing for a proper interpretation of LHC measurements [7,8]. In this thesis, we want to challenge two fundamental issues for the future of the LHC. The need for the simulators to match the collected data poses a computational challenge in the future. We propose fast generators trained on the expensive simulations to replace the bottleneck in LHC analysis. Secondly, direct searches of BSM physics from fully characterized theoretical models are limited by the sheer size of the parameter space. The new physics we are looking for might be hidden in a phase space corner still untouched by our current model-building approaches. A need of model agnostic searches is therefore needed to complement the standard analysis pipeline. Here, we look at observables which can boost anomaly searches in a fully data-driven way, i.e. without the need for simulations. Although seemingly far apart, we will present methods that bring together these two problems and seal them as two sides of the same coin from a machine learning point of view.

One development driving faster LHC simulations is the advent of deep generative networks. Such networks have shown great promise for LHC physics in the past few years, providing fast and accurate surrogates for simulations in high-dimensional phase spaces [9]. They learn the underlying probability distribution of events or calorimeter showers from a reference dataset and then generate new samples based on this learned distribution [10, 11] and we have already seen many successful applications on detector simulations [12–41]. For LHC physics, it is crucial that these networks are not used as black boxes, but their performance can be investigated, understood, and improved systematically [2, 42–46]. In the first part of this thesis, we will focus on the problem of building fast and accurate surrogate models for detector simulations. In practice, we consider the bottleneck in simulations, which is the calorimeter response. We approach the problem from two sides.

First, we will show how impressive speed and accuracy can be achieved with invertible neural networks (INN) [47–49], a particular version of normalizing flows (NF). Then, we switch focus and trade the speed for a more accurate description of the physics with a conditional flow matching (CFM) network for even higher dimensional feature spaces. However, there are also major challenges with scaling networks up to more granular (higher-dimensional) detectors [30, 31, 36, 41]. To alleviate the computational challenges, we also show how a lower-dimensional manifold can be learned by a variational autoencoder (VAE) and train a generative network in this lower dimensional latent space.

In the second part of the thesis we study the design of observables for beyond the standard model (BSM) searches reducing the assumptions on the specific physics model. Motivated by their initial success, ML methods for anomaly detection at the LHC were developed for tagging anomalous objects [50–59], anomalous events [60–78], or to enhance search strategies [79–87]. They include a first ATLAS analysis [88], experimental validation [89, 90], applications to heavy-ion collisions [91], the DarkMachines community challenge [92], and the LHC Olympics 2020 community challenge [93, 94]. The techniques used for this are strongly influenced by developments in modern machine learning. Autoencoders (AE) are simple tools for anomaly searches, based on a bottleneck in the mapping of a data representation onto itself. A better-suited definition is based on low-probability regions in the background phase space distributions [95–99]. However, the vanilla version of an AE is not a fail-proof proxy for the phase space density because of an inherent complexity bias. One of the goals of this thesis is to develop an autoencoder which is a robust anomalous tagger by defining a training strategy which provides a statistically well-defined density estimate as an anomaly score.

One issue with the density-based approaches [58, 100] is that the score is not invariant under simple transformations in the phase space. This means that a simple re-mapping of the momenta or coordinates fundamentally changes what the anomaly score is. This poses the question of how to choose a representation of the data for use in density-based anomaly detection tasks. Supervised machine learning methods use the idea of a truth label to optimise the neural networks, usually to classify between data with different truth labels. Unsupervised methods are those which do not require truth labels, instead optimising a network using a reconstruction loss or a negative log-likelihood, for example. Self-supervision on the other hand uses 'pseudo-labels', labels generated from the data without knowledge of a truth label, to optimise the networks. In contrastive learning [101], these labels correspond to a link between an original event and an augmented event. We define augmentations as some physical modification of the event kinematics.

Contrastive learning uses pseudo-labels to devise an auxiliary task for network optimisation through the contrastive loss function. Now the network learns how to process high-dimensional correlations in the data, and thus the representations learned by these networks can be very useful for downstream tasks. We introduce a pretraining strategy using contrastive learning tailored for density-based anomaly searches. It can be applied to objects reconstructed in an LHC event or to study the substructure in jets. The underlying idea is to learn a representation to which the underlying physics should be invariant, e.g. rotations, translations, or general transformations of the data. So the workflow is as follows; obtain a representation vector for each object in the dataset, then train an autoencoder on these representations to obtain the anomaly scores.

The thesis is structured as follows. We start with an introduction to the Standard Model of particle physics and collider physics in Ch. 2. The chapter follows with the role of simulations in collider phenomenology together with the description of the simulation chain. The last section gives a high-level overview of an analysis at the LHC. Ch. 3 provides all the necessary background to the ML techniques used in the later chapters, from the basics of neural networks to the specific implementation of network architectures. In Ch. 4 we discuss generative networks as fast surrogate models to replace expensive LHC simulations. The last section is dedicated to the evaluation of generative networks and their interpretability. The density-based approach to anomaly detection is discussed in Ch. 5. We present various robust anomaly scores tested on different datasets which are also invariant under specific transformations. Finally, the conclusions and the prospects are discussed in Ch. 6.

CHAPTER **2**

Physics at the LHC

Particle physics distinguishes from other physics areas for the profound understanding of the underlying theory. The Standard Model, given only a handful of parameters, has descriptive and predictive power. As a consequence, we have access to simulations unmatched by any other field. The precision and accuracy of our simulations allow for tests of the theoretical description of nature and the characterization of new physics effects.

This chapter concisely introduces the theory model used in particle physics and the colliders phenomenology. It starts with a general description of the Standard Model of particle physics, followed by the description of the physics in particle collisions together with their simulations. A dedicated section describes the interaction of particles with matter. The final section contains the general recipe for discoveries of effects not accounted for in the SM and future computational shortcomings that are challenged in the later chapters of this thesis.

2.1 Standard model of particle physics

The Standard Model of particle physics [102–104] is described in terms of quantum field theory [105], the theory that describes particles as relativistic quantum fields. The fundamental assumption in our description of nature is the global invariance under translations and Lorentz transformations, which defines the Poincaré group. In the SM, each fundamental particle is associated to an irreducible representation of the Poincaré group. The representation is characterized by its Casimir operators,

$$p^2 = M^2 \qquad \text{and} \qquad W^2 = M^2 S(S+1), \quad S = 0, 1/2, 1, 3/2 \dots , \tag{2.1}$$

which implies that the mass and the spin define the particle. For instance a massive scalar particle is defined by $S = 0$ and the mass M. We then specify the local symmetries of a particle by its invariance under a local gauge transformation. The corresponding gauge group is characterized by the algebra and its generators in different representations.

The SM follows a global invariance under the Poincaré group and a local invariance under the gauge group:

$$\mathrm{SU}(3) \times \mathrm{SU}(2)_\mathrm{L} \times \mathrm{U}(1) \;, \tag{2.2}$$

where SU(N) is the special unitary group of degree N. The SU(3) group describes quantum chromodynamics (QCD), or strong force. The $\mathrm{SU}(2)_\mathrm{L}\times\mathrm{U}(1)$ component encompasses the weak sector and quantum electrodynamics (QED). The interactions and the dynamic of fields is typically described in the Lagrangian formalism which manifestly shows the Lorentz invariance of the action. The global U(1) gauge invariance is promoted to a local invariance after introducing gauge fields. These are eight gluonic fields for SU(3) and four additional gauge bosons for $\mathrm{SU}(2)_\mathrm{L}\times\mathrm{U}(1)$, the photon, the W bosons, and the neutral Z boson.

The overall structure of the SM has three families of particles charged under SU(3), the quarks, and three families of particles interacting in the electroweak sector, the leptons. Then, there are the 12 Gauge spin-1 particles presented previously and, finally, the Higgs boson which gives mass to the W and Z bosons by spontaneous symmetry breaking, namely

$$\begin{aligned}
&\text{Quarks:} && \begin{pmatrix} u_i \\ d_i \end{pmatrix}_\mathrm{L}, u_R, d_R + \text{c.c} \\
&\text{Leptons:} && \begin{pmatrix} \nu_l \\ l \end{pmatrix}_\mathrm{L}, l_R, + \text{c.c.} \\
&\text{Gauge bosons:} && \gamma, W^{+/-}, Z, g, \\
&\text{Higgs boson:} && H
\end{aligned} \tag{2.3}$$

where $i = 1, 2, 3$ indicates the quark family, $l = e, \mu, \tau$ indicates the lepton family, and we do not explicitly write the charge conjugated particles. In the SM the full Lagrangian density of nature is summarized as

$$\mathcal{L}_{SM} = \mathcal{L}_\mathrm{gauge} + \mathcal{L}_\mathrm{fermions} + \mathcal{L}_\mathrm{Yukawa} + \mathcal{L}_\mathrm{Higgs} \tag{2.4}$$

This very compact representation includes all the particles observed until today and their interactions. This decomposition separates the dynamic of the fields from the interactions between them. The terms contain:

- $\mathcal{L}_\mathrm{gauge}$ includes the kinetic terms of the gauge bosons;
- $\mathcal{L}_\mathrm{fermions}$ includes the kinetic terms of fermions and the interaction with the gauge bosons;
- $\mathcal{L}_\mathrm{Yukawa}$, in this term are included the mass terms and the interaction of fermions with the Higgs boson;
- $\mathcal{L}_\mathrm{Higgs}$. Lastly, this term contains the dynamic description of the Higgs boson, its self-interaction, and the interaction with the gauge bosons.

The SM, as presented in Eq. 2.4, has 19 parameters. These are the 9 masses of the fermions, the Higgs boson mass, 4 parameters which describe the quark flavour mixing (CKM matrix), a strong CP-violation phase, 3 gauge couplings, and the Higgs vacuum expectation value.

Although this model has been providing a very precise description of the observed phenomena, it is known to not accommodate all the experimental observations. The

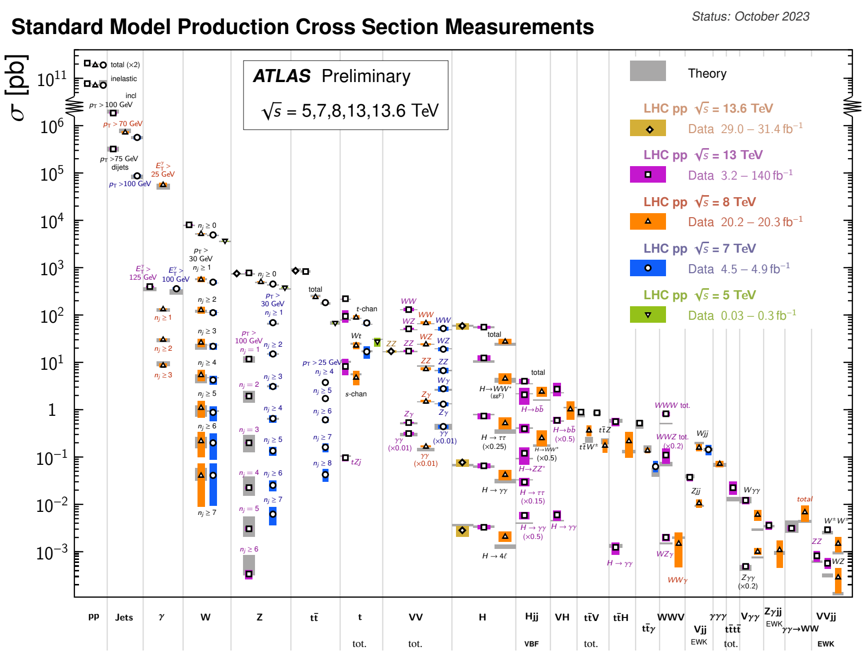


Figure 2.1: Summary of total and fiducial cross section measurements for different production channels at the ATLAS experiment [106].

observation of beyond the Standard Model effects can open the way to a better understanding of nature. Among the well-known open questions in the field, from astrophysical observations we detect the existence of dark matter and dark energy. These two add up to $\simeq 95\%$ of the total energy of the universe. However, the nature of dark matter has been eluding the direct and indirect searches and it is still unknown. The observation of neutrino masses is another missing piece of the SM in which neutrinos are assumed to be massless. The SM lacks a clear unique mechanism that gives mass to neutrinos. It is also known that the theory is not valid at the Planck energy scale anymore, where gravitation effects has to be taken into account. Depending on the point of view, naturalness and the hierarchy problem can be considered an additional issue of the SM which are studied in combination with the other open problems.

2.2 Simulations ex machina

This section describes the physics behind a collision in a hadron collider and the simulation of each step from the first-principle description of the hard scattering to the final interaction with the particle detector.

2.2.1 Hard scattering

The LHC has slowly turned into a precision machine, allowing for tests of the SM with exceptional precision. This is possible because, given the free parameters of the SM that have to be measured, the theory is predictive, i.e. it provides theory estimates of observables. For instance, Fig. 2.1 shows the cross section, namely the probability for a certain event to occur, for different processes in the SM. It is clear that the LHC has been providing high-precision measurements of the SM over many orders of magnitudes.

Taking a closer look, it stems from extremely precise theory predictions which is a rather unique scenario made possible by a controlled simulation chain.

Starting from the Lagrangian we can calculate a quantity related to the interaction strength, the cross section. The number of expected events is proportional to the cross section and thus of primary importance for colliders. Its calculation involves solving a high-dimensional integrals that might not be easy to compute in practice. Given two incoming particles and n final states, the differential kinematic phase space region is written as

$$\mathrm{d}\Phi = (2\pi)^4 \prod_{i=1}^{n} \frac{\mathrm{d}^3 p_i}{(2\pi)^3} \frac{1}{2E_i} \, \delta^{(4)}(p_1 + p_2 - \sum_i p_i), \tag{2.5}$$

where $d\Phi$ is the phase space element for on-shell incoming particles p_1, p_2 and outgoing particles $p_i, \ldots, p_n$. The delta function ensures momentum conservation in the scattering process. The (squared) matrix element $|\bar{M}^2|$ contains the information calculable from first principles in QFT and it expresses the phase space weights of the considered process as a function of a scalar combination of the four momenta. These pieces are combined to obtain the differential cross section from

$$d\sigma = \frac{1}{F} |\bar{M}^2| d\Phi, \tag{2.6}$$

and the total cross section as

$$\sigma = \int d\Phi \frac{d\sigma}{d\Phi} \tag{2.7}$$

where F is the flux factor, the modulo of the difference between the velocities of the two beams. The generation of events from the hard scattering process is done by sampling, with Monte Carlo methods, from the normalized differential cross section. Among the methods that efficiently sample the phase space, we have MG5aMC [107], Pythia8 [108], and Sherpa [109].

In a proton-proton collider, the scattering is well-described by the collision of two constituents inside the protons, each one carrying a fraction x of the total momentum. Given the two interacting partons a and b, the available center-of-mass energy is rescaled as

$$\bar{s} = x_a x_b s, \tag{2.8}$$

with the center-of-mass energy s defined by the momentum of the two protons. The probability of having a particular parton involved in the collision is not analytically calculable from QFT and it is measured from data. This is called a parton distribution function (PDF) $f(x, Q^2)$ and modifies the cross section for the production of the final state X at the energy scale Q^2 as

$$\sigma_X = \sum_{a,b} \int_0^1 dx_a dx_b f_a(x_a, Q^2) f_b(x_b, Q^2) \sigma_{ab \to X}. \tag{2.9}$$

This equation convolves the cross section with the parton distribution function of the proton $f_i(x_i, Q^2)$, summed over all the partons. The most outstanding methodolgy that extracts these functions is NNPDF [110] which uses machine learning for accurate predictions.

2.2.2 QCD effects

The full description of a QCD hard scattering process is more involved. The calculation of any cross section has to take into account the emission of additional particles. This is a consequence of collinear singularities appearing in any calculation with additional final state quarks or gluons. A general factorization equation for a gluon collinear to a quark holds [111],

$$d\sigma(X \to Y + g) = d\sigma(X \to Y) dx dz \frac{1}{x} \left[\frac{\alpha_s}{2\pi} C_F \frac{1+z^2}{1-z} + \mathcal{O}(\frac{x}{Q^2}) \right] . \tag{2.10}$$

This expression basically says that in the collinear limit we cannot distinguish between the final state Y and the final state $Y + g$. The form of the differential cross section includes a collinear term, written as a function of a variable x singular in the collinear limit, and a splitting kernel that depends on the fraction of momentum z carried by the additional particle. This function is one of the QCD splitting kernel,

$$P_{qq}(z) = \frac{1+z^2}{1-z} \tag{2.11}$$

These splitting kernels are universal and are also used in the DGLAP equations to calculate the evolution of the parton distribution functions from the reference scale to the one of interest.

As a final result, the hard scattering process is accompanied by radiation of particles in the initial state (ISR) and in the final state (FSR). These effects are often considered noise that complicates the observation of the hard scattering process. The simulation of ISR and FSR exploits this description of a sequence of splittings to generate a parton shower. The splitting kernel represents the probability for an off-shell q/g emission. Given a hard scale Q^2, the current energy scale t, and the allowed splitting fraction z', z'', the probability of finding a new particle is

$$P(t) \propto \int_{z'(t,Q^2)}^{z''(t,Q^2)} P_{ab}(z) dz. \tag{2.12}$$

This equation is used in simulators to calculate the probability of the parton not undergoing a splitting process between the two scales t', t'', also known as the Sudakov factor:

$$\Delta_a(t', t'') = \exp\left\{ - \sum_{b \in q,g} \int_{t'}^{t''} \frac{dt}{t} \int_{z'}^{z''} \frac{\alpha_S}{4\pi} P_{ab} \right\} . \tag{2.13}$$

Various implementation of event generators, like Pythia8 [108], Sherpa [109], and Herwig [112], use this description for the parton showers.

However, particles from QCD cannot be observed directly at our energy scale. This is a direct consequence of the non-abelian structure of QCD [105, 111]. This effect is also referred to as confinement, i.e. free fundamental particles can only be observed at high energies. In other words, perturbative QCD only works at high energy and we have to deal with large non-perturbative effects at the scale $\Lambda_{\text{QCD}} \simeq 1\,\text{GeV}$. The observed final state is a collimated collection of hadrons referred to as "jets".

Empirical models explain this last non-perturbative step, also called "hadronization". The hadronization process starts from assumptions on the non-perturbative interaction between quarks. The most used methods are the Lund string model [113] and the cluster hadronization model [114]. As an example, in the Lund string model the potential between quarks increases linearly $V(r) \approx kr$ with the distance and quarks are pushed further away. The increasing energy breaks the "string" causing a split which, by color confinement, produces new color neutral particles. The process is algorithmically implemented in PYTHIA8 [108] in a sequantial way. It starts from the initial selection of the size of the string, to the flavour selection, the p_T, and the energy sampling. This short descriptions serves to remark that these models have several parameters that have to be tuned. However, a global tuning that fits the data is not possible, which implies a possible model dependence in studies highly dependent on the hadronization model. We introduce an example in Sec. 2.3, presenting data-driven strategies that avoid simulation biases.

2.2.3 Interaction with matter

All the particles produced during the process are finally reconstructed by detectors located around the interaction point. The end-to-end simulation chain also includes this step and it is largely tied to the interaction of particles with matter. This section introduces the physics processes in the detector and their simulations, while details on the detector structure are discussed in Sec. 2.3. Only particles with a long lifetime reach the detector and interact with matter according to different processes. Among the particle zoo, eight are the most frequently observed: $e^{+/-}, \mu^{+/-}, \gamma, \pi^{+/-/0}, K^{+/-}, K^0, p^{+/-}, n$. The interaction of these particles is grouped as:

- charged electromagnetic interactions: e, μ interact by exciting the material or by emission of radiation;
- photon interactions: photons undergo photo-electric effect, Compton scattering, or pair production;
- hadronic interactions: all the other hadrons can directly interact with the nucleus as well as incur in inelastic scattering. This makes the number of possible interaction large and difficult to model.

Charged electromagnetic interactions are well described by the Bethe-Bloch formula [115]. It relates the energy loss per unit length dE/dx as a function of the momentum p or, more often, of the product $\beta\gamma = p/(Mc)$. Given a material with atomic number Z, nuclear number A, charge z, and minimum ionisation potential I, the Bethe-Bloch [116] function is

$$\left\langle \frac{dE}{dx} \right\rangle \propto z^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\ln \frac{1}{2} \frac{2m_e(c\beta\gamma)^2 W_{\max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right] , \tag{2.14}$$

where m_e is the electron mass, $W_{\max}$ is the maximum energy transfer by the particle, and δ is a corrective function for high-relativistic radiative effects. From this equation we distinguish three regimes. At high energy electrons and muons interact via Bremsstrahlung while at lower energies the interaction is dominated by ionization/excitation. An example of the Bethe-Bloch function is shown in Fig. 2.2 for muons traversing Copper.

The energy loss for photons interacting with matter is equally understood and well modeled. As a function of the energy, the energy loss is characterized by primarily three interaction modes. At low energy the cross section is dominated by the photoelectric effect, i.e. emission of electrons from the material after the absorption of the incoming photons. At higher energies the incoherent scattering off an electron, or Compton

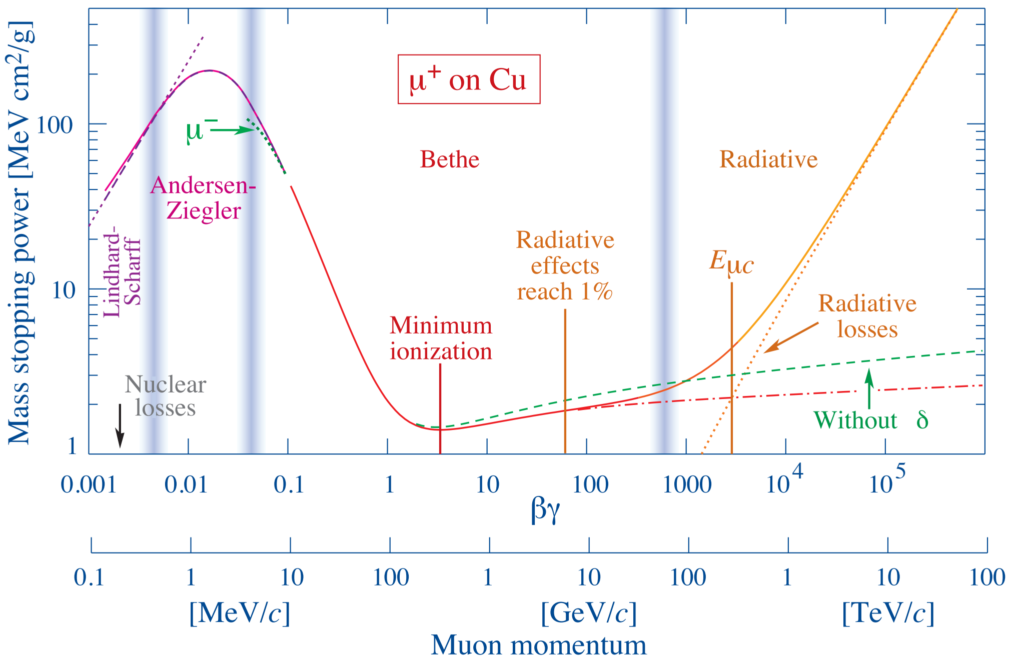


Figure 2.2: Bethe-Bloch function for a positively charged muon interacting with Copper [115]. The mean energy deposition $\mathrm{d}E/\mathrm{d}x$ in the material divided by the density gives the mass stopping power here showed as a function of the energy of the incoming particle

scattering, prevails until the energy threshold of ~ 1 MeV where the production of an e^+e^- pair is possible and quickly takes over the total cross section. The exact dependence of the cross section as a function of the energy varies with the material. An example for $Z = 6$ carbon is shown in Fig. 2.3. Notice that in these interactions, except for Compton scattering, photons are absorbed hence the interactions are presented in terms of cross sections instead of energy deposition per unit-length.

Hadrons interactions are more complicated to model because of the vast and complex range of nuclear interactions that have to be taken into account. A high-energetic interacting hadron produces charged particles and neutrons at different times and in a wide energy spectrum of about $1\text{eV} \lesssim E \lesssim 1\text{GeV}$. Additionally, the emission of photons introduces an electromagnetic component in the chain of interactions. The interest of this thesis lays in hadrons of $\mathcal{O}(1)$ GeV energy. These particles collide with the nuclei and likely strongly interact causing nuclear spallation, which cannot be treated by perturbative QCD. Initially, the incoming hadrons deeply penetrate the nucleus and induce a nuclear cascade which produces final-state and excited nucleons, charged and neutral pions, and kaons. A subset of these particles are emitted and keep interacting with the material while others are absorbed. In a second phase the excited nucleus emits neutrons, photons, and α particles. High-energetic neutrons mostly undergo elastic and inelastic scattering. In the latter case, the recoil nucleus, after excitation, emits photons. Neutrons can also be absorbed by the nucleus with detectable secondary radiation. This radiation comprises charged particles, photons, neutrons, and even fission products in materials with high Z. These effects are explained from a combination of nuclear physics and collected data.

The detailed modeling of electromagnetic and hadronic interactions pertinent to this thesis is described in the next section. The simulation of all these interactions with

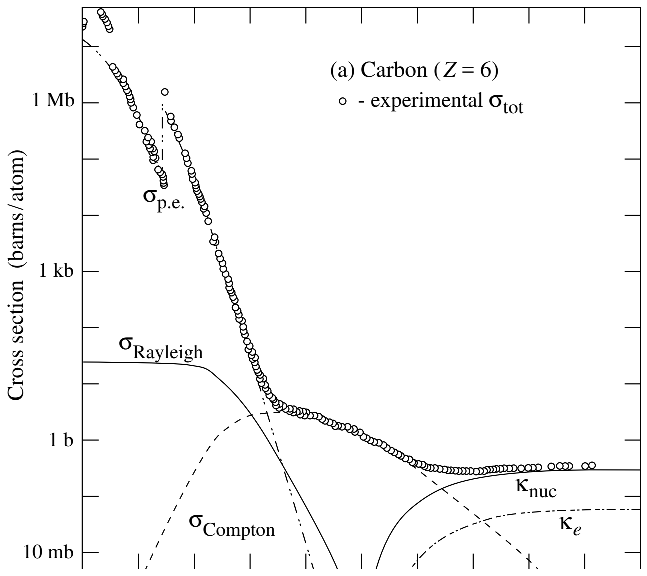


Figure 2.3: Cross section per atom of photons interacting with Carbon as a function of the energy of the photon [115].

matter is done again following Monte Carlo techniques. The most used tool in HEP is GEANT4 [117–119], which is a flexible framework for the simulation of any type of interaction with matter. GEANT4 allows for a user-defined detector geometry and list of interactions that have to be taken into account. Then, given the nuclear information of the material, the simulator keeps track of all the particles produced in the material together with their space-time history. As the energy and, consequentially, the number of particles grow, the simulation in GEANT4 can become considerably computationally intensive. Therefore, detector simulations are considered the limiting factor in the simulation chain.

2.2.4 Calorimeter showers

The typical quantities of interest for particles produced from the hard scattering are the position and the energy. The measurement of the energy is a destructive process where the energy is progressively deposited in the material and detected. However, at the energy frontier, this process is not immediate and new particles are produced from the interaction. Starting from the electromagnetic interactions, at energies $E \geq 1\text{GeV}$ the main interaction mechanisms are Bremsstrahlung and pair production. From these two processes, it is possible to define a simple model for electromagnetic interactions. An useful quantity used to describe the passage of electrons through matter is the radiation length X_0, defined from nuclear properties of the material and therefore function of Z, A, and dimensional constants [120].

The radiation length provides the mean distance needed for an electron to reduce its initial energy by a factor e. For colliders, the definition of X_0 is often reformulated in an approximate way as

$$X_0 = \frac{(4\alpha r_e^2 N_A/A)^{-1}}{Z(Z+1)\ln(287/\sqrt{Z})} \tag{2.15}$$

where we defined the fine structure constant α, the electron radius r_e, and Avogadro's number N_A. For simplicity, these constants are usually replaced by their numerical

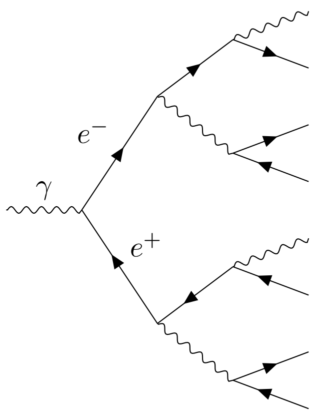


Figure 2.4: Schematic of an electromagnetic shower.

product as $(4\alpha r_e^2 * N_A/A)^{-1} = 716.4$ g cm^{-2}. Following this construction the average energy loss due to Bremsstrahlung is

$$\left\langle \frac{dE}{dx} \right\rangle = -\frac{E}{X_0} \tag{2.16}$$

and after integrating over the thickness x of the material,

$$\langle\, E(x)\,\rangle = E_0\, e^{-x/X_0} \tag{2.17}$$

where E_0 is the initial energy. The same quantity is retrieved in the calculation of the scattering cross section for photons undergoing pair production. The cross section for high-energetic photons is written as [115]

$$\sigma_{\text{pair}} = \frac{7}{9}\frac{A}{N_A}\frac{1}{X_0}, \tag{2.18}$$

which can be used to described the number of photons in a beam after penetrating the material as a function of the position x. For an initial number of photons N_0, the number of photons in the beam can be described by

$$N_\gamma(x) = N_0 \exp -\frac{7}{9}\frac{x}{X_0}\,. \tag{2.19}$$

Eq. 2.19 shows that the number of photons is reduced by a factor of e after one interaction length. From these observations, we expect photons and electrons/positrons to behave similarly when interacting with matter in the high energy regime, except for small differences originating from the different nature of the two interactions.

Given these two processes, a simple model for the deposition of energy in a block of material consists of assuming each particle interacts after $1X_0$. Following the example in Fig. 2.4, the originating particle initiates a so-called shower, a cascade of particles of progressively lower energy. In this simple model, the number of particles at time $t(X_0)$ is

$$N(t) = 2^t \tag{2.20}$$

and, assuming an equal splitting of the energy between the two final state particles, the energy at the same step is

$$E(t) = E_0/2^t\,. \tag{2.21}$$

The showering process continues until the particles reach a critical energy E_c after which everything is absorbed by either ionisation, Compton, or photoelectric effect. Formally, the critical energy is defined by the point where the ionization and the radiative energy loss match:

$$\left(\frac{dE}{dx}\right)_{\text{ionization}} = \left(\frac{dE}{dx}\right)_{\text{radiative}} \tag{2.22}$$

A characteristic of these showers is the typical scaling of the space needed to contain the shower longitudinally, which in our simple model is

$$t_{\text{max}} = \frac{\ln(E_0/E_c)}{\ln 2} \tag{2.23}$$

Although more realistic simulations describe the data better, the $\log(E)$ scaling is approximately respected which further motivates the usage of energy detectors via the destructive measurement of the full energy deposition. These instruments are called calorimeters and further discussed in Sec. 2.3.

Regarding the hadronic interactions, a similar showering process occur inside the material. However, as already discussed in the previous section, the complexity of the strong interactions does not allow for a similar theoretically motivated description. It is possible to define the mean free path λ for a proton or neutron to undergo a nuclear interaction. Starting from the geometric cross section of a proton interaction with a material with atomic number A,

$$\sigma \propto A^{1/3} \tag{2.24}$$

The mean free path is defined as

$$\lambda(\text{ g cm}^2) = 35 \text{ g cm}^2 A^{1/3} \tag{2.25}$$

Hadronic showers are broader and longer than electromagnetic showers due to the typical larger mean free path. They include an electromagnetic component from photons radiated from the nucleus or from π_0 decay. Because of the plethora of processes in hadronic interactions with matter, as described in Sec. 2.2, hadronic showers are modeled with parametric functions fitted to data.

2.3 Discovering New Physics

In this section, we introduce the general structure of high-energy detectors, the workflow for the reconstruction and triggering of events, and the summary of the tasks we will focus on in the rest of the thesis.

2.3.1 Particle detectors

Detectors at the LHC have varying design. The multi-purpose detectors are ATLAS and CMS, which aim to generally test aspects of the SM and search for BSM effects. Two other experiments focus on the b quark physics, LHCb, and heavy ion physics, ALICE, respectively. The structure of a multi-purpose detector, like ATLAS and CMS, has a cylindrical shape with full coverage in the azimuthal direction immersed in a magnetic field. Trackers are the first elements in the detector. Charged particles interact with the active material leaving tracks used to reconstruct the transverse momentum and the position of the interaction point. Additionally, the magnetic field curves the tracks providing information on the particle charge. The next shell detects the energy of the incoming particles using calorimeters. In the destructive interaction with the material, particles produce a "calorimeter shower" while depositing energy as described in the previous section. For several reasons, including cost and general needs, the hadronic calorimeter (HCal) is separated from the electromagnetic (ECal), therefore resulting in two calorimetry systems specialized in fully containing both showers. In practice, the hadrons showering often starts in the ECal leaving a signature that can be matched to an energy deposition in the HCal. At this stage the only stable particles are muons which deposit a small amount of energy in the calorimeters and continue towards the muon chamber. In this final shell, the muon momenta is inferred from the interaction with the active material.

2.3.2 Reconstruction and triggering

The collection of the information from the different parts of the detector is the key to high-quality reconstruction of the particles of a collision event. A crucial step in the analysis chain consist of summarizing the information contained in the $\mathcal{O}(10000)$ readout channels into a handful of particles that originate from the interaction point. An electron is defined by a charged track matched to an energy cluster in the ECal. An energy cluster in the ECal, without any associated track, defines a photon. Both selections also have to fulfill an isolation criterion. The collimated spray of final-state particles coming from the showering and hadronization of colour charged particles is reconstructed as a "jet". A jet requires energy deposition in both the ECal and the HCal, however the definition of the constituents inside the jet depends on the reconstruction algorithm. Sequential clustering algorithms are often used for jets reconstruction. These algorithms define a distance measure between constituents and iteratively group them under a distance condition. The distance is calculated, given a radius parameter R, as

$$d_{ij} = \min(p_{Ti}^n, p_{Tj}^n)\frac{R_{ij}^2}{R^2} \tag{2.26}$$

$$d_{iB} = p_{Ti}^n \tag{2.27}$$

where n is defined by the clustering algorithms, and d_{iB} is the p_T distance from the beam axis. Typical choices are $n = 0, 2, -2$, respectively the Cambridge/Aachen, k_T, and anti-k_T algorithms. The distance between constituents, defined as R_{ij}, is calculated in the (η, ϕ) plane as

$$R_{ij}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2 \,. \tag{2.28}$$

The two constituents are clustered if $d_{ij} < d_{iB}$. This process continues until all particles are clustered or a fixed number of jets is found. Finally, a charged track in the muon chamber defines a muon. This is usually a clean signature because all the other particles have already been stopped by the detector. The combined, more refined, information from the entire detector improves the reconstruction of candidates. An example of such approach is Particle Flow [121, 122], used by CMS and ATLAS. After reconstructing all the objects, it is possible to define the amount of missing energy from energy-momentum conservation and its direction. The presence of neutrinos, which do not interact with the detector, can be inferred from the missing energy as well.

The rate at which collision happen of 40 MHz poses constraints on the timing for the online reconstruction of events as well as the overall number of stored events.

Therefore, the storage system at the LHC is provided with a multi-level triggering system. Considering the CMS experiment as an example, the first trigger system (L1) reduces the output rate from 40 MHz to $\sim$ 100 kHz. This system is hardware-based and it performs a selection of interesting events based on an hand-crafted triggering menu. The event composed of coarsely reconstructed electrons, muons, and jets passes the trigger if it fulfills particular kinematics or multiplicity conditions. The second trigger (HLT) is software-based and exploits all the information, including tracking information, collected in the detector for a more precise reconstruction. Here, the data stream is further reduced from $\sim$100 kHz to about 1 kHz and then saved for future analysis.

2.3.3 Physics analysis

One of the achievements of the LHC is the discovery of the Higgs boson in 2012 [123,124]. The procedure from collected raw data to the final hypothesis test is quite involved and requires a deep understanding of the experimental conditions, the simulations, and the underlying physics. In a standard analysis, a signal hypothesis is formulated based on model assumptions. Then, the search space is optimized to maximize the ratio between signal and background. This phase includes defining the fiducial region with kinematic cuts restricting to the interesting phase space regions with well-behaving detector response. Accurate simulations of the BSM scenario are compared to the collected data in the selected observables and, in the frequentist approach, a likelihood-ratio test provides the summary statistic for the background only H_0 versus background plus signal H_1 hypothesis. If the alternative hypothesis is rejected, the analysis provides limits on parameters of the signal model, e.g. the mass of a new particle. The extraordinary accurate modeling of the SM, together with precise measurements, allows not only for searches of new particles but also for consistency checks of our understanding of the SM.

However, at different levels there are assumptions which can obstacle the observation of BSM effects. Already at the L1 trigger level the hand-crafted menu might be limited by what physicist find interesting and completely mask out BSM physics. Similarly, the number of models which people can design is limited compared to Nature and the answer might be that we are looking in the wrong place. Both aspects advocate for more model agnostic approaches that sacrifice sensitivity to specific signal in favor of coverage. Having access to data, this can be done in a fully data-driven way without any dependence on differences between data and simulations.

2.3.4 Goals

This thesis proposes to enhance aspects of analysis at colliders. Looking at the future of the LHC, the increase in luminosity for the High Luminosity-LHC (HL-LHC) will require a steep increase in the number of simulations which comes with an increase in computational costs which are currently above the budget. Simulating the detector response is the most expensive part of the simulation and Ch. 4 discusses how to provide fast, accurate, and precise detector simulations. Regarding model agnostic searches, Ch. 5 includes results on unsupervised ML methods for data-driven searches and techniques to define powerful observables that respect symmetries of the data. The next chapter shows how these two problems are brought together by machine learning and can be formulated as similar tasks, i.e. density estimation of high-dimensional phase spaces.

CHAPTER **3**

Machine learning

The large amount of data available at the LHC opens the avenue to the development of methods and tools based on machine learning techniques. Paired with powerful simulators which provide labeled data, we have an ideal framework for ML to shine. Indeed the usage of ML is becoming ubiquitous in HEP, from classification of jets [125, 126], to unsupervised searches [127, 128], and applications of generative networks [11, 129].

This section provides some background on the neural networks (NN) used in this thesis. Sec. 3.1 starts with the training strategy of these networks. Then, in Sec. 3.2 we describe the details of three applications of modern machine learning, i.e. classification, generation, and representation learning. Finally, we focus on the specific architectures in Sec. 3.3.

3.1 Deep learning basics

From our point of view, a NN is a high-dimensional parameterization which can represent a large family of functions, and the training process is nothing more than a procedure to find the best fitting parameters. The fundamental units, the "neurons", constitute the layers in the NN and different implicit biases can be enforced by the specific architecture. Here, we start with the basics of defining a NN and its training methodology.

3.1.1 Fully connected networks

The simplest building block of many networks is a fully connected layers. Given an input $x \in \mathbb{R}^{d_x}$, the layer applies a linear transformation with a weight matrix $W \in \mathbb{R}^{d_x \times d_y}$ and a bias vector $b \in \mathbb{R}^{d_y}$ as learnable parameters to give the output $y \in \mathbb{R}^{d_y}$. The extension to more complex functions follows from the introduction of a non-linearity σ applied to the output as

$$y = \sigma(Wx + b)\,. \tag{3.1}$$

It has been shown that a neural network with a hidden layer of infinite width can approximate any continuous function [130, 131]. Typical choices for the non-linear function are part of the Rectified Linear Unit family, e.g.

$$\text{ReLU} = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} , \tag{3.2}$$

$$\text{LeakyReLU} = \begin{cases} x & x > 0 \\ \alpha x & x \leq 0 \end{cases} \tag{3.3}$$

$$\text{Swish} = x\,\sigma(x) \tag{3.4}$$

where σ is the sigmoid function

$$\sigma(x) = \frac{1}{1+e^{-x}} . \tag{3.5}$$

There are cases where we prefer to have a bounded output from the network, typically to have a probabilistic interpretation of the output. In these scenarios a typical choice is the sigmoid function defined in Eq. (3.5) or a rescaled version of it.

3.1.2 Loss function and training

Once we define the network that approximates the function, we need to define a strategy to optimize its parameters. The common approach leverages highly efficient packages for gradients calculations to minimize a loss function with respect to the network parameters. The choice of the loss depends on the specific problem. As an example, suppose we have a regression problem where, given a set of inputs x, we want to predict a single output $f(x)$. We approximate the function with a neural network $f_\theta(x)$ with learnable parameters θ. A naive loss function for this problem is the mean squared error (MSE) loss

$$\mathcal{L} = \Big\langle ||f(x) - f_\theta(x)||_{L2} \Big\rangle_{p_\text{data}} . \tag{3.6}$$

The loss function in Eq. (3.6) is estimated on the phase space points by using training data that describes the function $f(x)$, i.e. we have labeled data $(x, f(x))$ available. During training the loss is evaluated sampling from the training distribution p_data. The training that makes use of labels is called “supervised”, while the completely label-free counterpart is called “unsupervised”.

The backpropagation algorithm [132] allows for the computation of the derivative of the loss with respect to all the network parameters θ. With this information the network is updated, from step t to $t+1$, in the direction that minimizes the loss

$$\theta_i^{t+1} = \theta_i^t - \lambda \frac{\partial \mathcal{L}}{\partial \theta_i^t} . \tag{3.7}$$

The parameter λ is called “learning rate” and, in its simplest form, is a scalar value that changes the size of the update step. In more complicated training algorithms, λ is not constant during training and it differs for each weight. An example is the OneCycleLR [133] where, after a warm-up period, λ follows a cosine annealing policy. The Gradient Descent (GD) method in Eq. (3.7) is a relatively standard technique which has been improved in more recent methods, like the Adam [134] or the AdamW [135].

However, this minimization procedure can end up in one of the local minima of the loss landscape and provide a sub-optimal solution. A practical solution to this problem is to introduce a noisy version of the update that can more easily explore the loss landscape. A stochastic version of the update rule, which considers M batches of the training dataset, can ameliorate this problem. Therefore, we split the loss as

$$\mathcal{L}_{\text{batches}} = \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}_i \,. \tag{3.8}$$

and update the network parameters after each batch.

The next section describes three particular problems that can be solved by ML. Meanwhile we also describe a theoretical founded approach to derive an optimal loss function.

3.2 The Mexican standoff

There is a common categorization of neural networks based on their outputs. In a classification problem the output is a probability score, while in a generative task the network provides a density estimate as well as the possibility of sampling new datapoints from the phase space distribution. A third aspect is the automated learning of new data representations which are compact and more data efficient. We consider the unsupervised learning of the data manifold and the explicit learning of known symmetries as an example.

However, the distinction between these classes is fluid and more involved methods may combine the different approaches. Here, we introduce the theory of the networks that are used in Ch. 4 and Ch. 5.

3.2.1 Classification

For pedagogical reasons, we start by describing classifiers. Suppose we have two probability distributions $p_0(x)$ and $p_1(x)$ and a dataset X. A two-hypothesis test has a null hypothesis H_0, which states that the data is sampled from the distribution p_0 and the alternative H_1, which contrarily assumes that data is sampled from p_1. For a simple hypothesis the best summary statistic exists and it is the likelihood ratio

$$L(X) = \frac{p_0(X)}{p_1(X)} \tag{3.9}$$

Formally, the Neyman-Pearson lemma [136] proves that, given a significance level α for H_0, the likelihood ratio is the most powerful test statistic. This means that the test maximizes the probability of rejecting H_0 when H_1 is true, i.e. its power.

By an accurate choice of loss function, we can approximate $L(x)$ with a neural network $C(x)$. Given the two distributions p_0 and p_1, and the labels $y \in \{0, 1\}$, the minimization of the binary cross-entropy loss function

$$\mathcal{L}_{\text{cls}} = -\langle \log C(x) \rangle_{p_0} - \langle \log(1 - C(x)) \rangle_{p_1} \tag{3.10}$$

provides a NN which converges to the likelihood ratio. This can be easily derived using functional derivatives. We can calculate the exact minimum of the loss as

$$\begin{aligned}
0 &\overset{!}{=} \frac{\delta \mathcal{L}}{\delta C} && (3.11)\\
&= \frac{p_0}{C} - \frac{p_1}{1-C} && (3.12)\\
&= (1-C)p_0 - Cp_1 && (3.13)\\
\Rightarrow C(x) &= \frac{p_0}{p_0+p_1}(x) && (3.14)
\end{aligned}$$

where we omitted the x dependence. From Eq. (3.14) we can define a weight per phase space point as

$$w(x) = \frac{p_0}{p_1}(x) = \frac{C}{1-C}(x) \tag{3.15}$$

In Sec. 4.5 we will see the importance of this quantity to reweight distribution and, in particular, to the systematic understanding of generative networks.

The binary cross-entropy loss is implemented by applying the sigmoid function defined in Eq. (3.5) to the final layer of the network. Other choices are possible [137] but, although they should converge to the same quantity, are less numerically stable.

3.2.2 Generation

Unlike classification, the aim of a generative network is to reproduce the distribution p_{data} encoded in the training data. For instance, if the data is the result of a forward model, the generative network will be a surrogate model for that forward process. The two approaches of interest involve estimating the density of the distribution. This is useful not only for generating new data but also for studying out-of-distribution (OOD) detection. We distinguish between networks which define a transformation between two distributions, we refer to these as “flow networks”, and cases where the network directly predicts a likelihood.

The training has foundations in statistical physics. The ideal loss function for this task would be a measure of the distance between the network and the data distribution that is zero when the two distributions match exactly. The f-divergence $f(t) = t\log t$, although not a distance measure in the mathematical sense, is often the basic recipe for a loss function, the Kullback-Leibler (KL) divergence. The KL-divergence is defined as

$$\text{KL}(p_{\text{data}}(x)|p_\theta(x)) = \int p_{\text{data}}(x) \log\left(\frac{p_{\text{data}}(x)}{p_\theta(x)}\right) \text{d}x\,, \tag{3.16}$$

which is positive definite and zero if $p_\theta = p_{\text{data}}$. Starting from Eq. (3.16), we can derive the negative log-likelihood (NLL) loss function:

$$\begin{aligned}
\mathcal{L} &= \text{KL}(p_{\text{data}}(x)|p_\theta(x)) && (3.17)\\
&= \int \text{d}x\, p_{\text{data}}(x)\log p_{\text{data}}(x) - \int dx\, p_{\text{data}}(x) \log p_\theta(x) && (3.18)\\
&= -\int \text{d}x p_{\text{data}}(x)\log p_\theta(x) + c && (3.19)\\
&= -\langle\, \log p_\theta(x)\,\rangle_{p_{\text{data}}} + c\,, && (3.20)
\end{aligned}$$

where c is a constant that does not depend on the network parameters θ. An unbiased estimator of the loss is the Monte Carlo average, i.e.

$$\mathcal{L} \approx -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(x_i)\,, \tag{3.21}$$

with N random samples from the training dataset and where we drop the constant term c.

Flows

The flows used in this thesis are mappings between two spaces with the same dimensionality. The important equation for flow architectures is the change of variable formula. Defining the invertible transformation G and its inverse $\bar{G}$ as

$$G_\theta(x) : x \to z \tag{3.22}$$
$$\bar{G}_\theta(z) : z \to x \qquad x, z \in \mathbb{R}^d\,, \tag{3.23}$$

the change of variable formula defines a bijective mapping from the data space x to the latent space z as as

$$p_{\text{lat}}(z) = p_\theta(G_\theta(x))\left|\frac{\partial G_\theta(z)}{\partial z}\right| \tag{3.24}$$

If we invert Eq. (3.24) and we take the negative logarithm of the network distribution, we obtain again the KL loss of Eq. (3.20). Then, the definition of the flow loss function is the negative log-likelihood

$$\begin{aligned}\mathcal{L}_\text{F} &= -\langle\, \log p_\theta(x)\,\rangle_{p_\text{data}} \\ &= -\Big\langle\, \log p_\text{lat}(\bar{G}(x)) + \log\left|\frac{\partial \bar{G}}{\partial x}\right|\,\Big\rangle_{p_\text{data}}\,.\end{aligned} \tag{3.25}$$

For a standard Gaussian latent distribution the first term of Eq. (3.25) is

$$-\log p_\text{lat}(z) = \log(\sqrt{2\pi}) + \frac{z^2}{2}\,. \tag{3.26}$$

As showed in the previous section, the loss in Eq. (3.25) is minimized if $p_\theta = p_\text{data}$. After training, when the approximate solution is found, the network gives access to a way of estimating the likelihood via the forward pass. If we choose a tractable latent distribution like in Eq. (3.26), we can easily generate new samples by sampling z from the latent space distribution $p_\text{lat}(z)$ and use the inverse mapping to obtain new samples in the data space. The modeling of the data distribution is extended to the conditional case by introducing the additional information as a condition to the transformation:

$$\mathcal{L}_\text{cF} = -\langle \log p_\text{lat}(\bar{G}(x;c)) + \log\left|\frac{\partial \bar{G}(x;c)}{\partial x}\right|\rangle_{p_\text{data}}\,. \tag{3.27}$$

The simple formalism of such neural network faces a few constraints. First, the Jacobian and its inverse have to exist, meaning that only invertible transformations are applicable. Second, the computation of the Jacobian has to be fast enough to allow multiple model evaluations resulting in a reasonable training time. This limits the choices in the architecture and introduces a trade-off between expressivity and speed that has to be taken into account. The implementation details of these networks are discussed in Sec. 3.3.

Energy-based networks

Energy-based models (EBM) are a class of probability density estimation models appealing for their flexibility. They are defined through a normalizable energy function, which is minimized during training. This energy function can be chosen as any non-linear function mapping a point to a scalar value [138],

$$E_\theta(x):\ \mathbb{R}^D \to \mathbb{R}\ , \tag{3.28}$$

where D is the dimensionality of the phase space. The EBM uses this energy function to define a probabilistic loss, assuming a Boltzmann or Gibbs distribution as its probability density over phase space,

$$p_\theta(x) = \frac{e^{-E_\theta(x)}}{Z_\theta} \qquad \text{with} \qquad Z_\theta = \int_x \mathrm{d}x e^{-E_\theta(x)}\ , \tag{3.29}$$

where we defined the partition function Z_θ. We omit an explicit scaling of the energy by a temperature or some other constant in this formula. The main feature of a Boltzmann distribution is that low-energy states have the highest probability. The EBM loss is the negative logarithmic probability evaluated as a likelihood over the model parameters,

$$\begin{aligned} \mathcal{L}(x) &= -\log p_\theta(x) = E_\theta(x) + \log Z_\theta \\ \Rightarrow \qquad & \mathcal{L} = \langle E_\theta(x) + \log Z_\theta \rangle_{p_\text{data}}\ , \end{aligned} \tag{3.30}$$

where we define the total loss as the expectation over the per-sample loss. The difference to typical likelihood losses is that the second, normalization term is unknown.

To train the network we want to minimize the loss in Eq. (3.30), so we have to compute its gradient,

$$\begin{aligned} \nabla_\theta \mathcal{L}(x) &= -\nabla_\theta \log p_\theta(x) \qquad (3.31) \\ &= \nabla_\theta E_\theta(x) + \nabla_\theta \log Z_\theta \\ &= \nabla_\theta E_\theta(x) + \frac{1}{Z_\theta} \nabla_\theta \int_x \mathrm{d}x e^{-E_\theta(x)} \\ &= \nabla_\theta E_\theta(x) - \int_x \mathrm{d}x \frac{e^{-E_\theta(x)}}{Z_\theta} \nabla_\theta E_\theta(x) \\ &= \nabla_\theta E_\theta(x) - \Big\langle \nabla_\theta E_\theta(x) \Big\rangle_{x\sim p_\theta}\ . \qquad (3.32) \end{aligned}$$

The first term in this expression can be obtained using automatic differentiation from the training sample, while the second term is intractable and must be approximated. Computing the expectation value over $p_\text{data}(x)$ allows us to rewrite the gradient of the loss as the difference of two energy gradients

$$\begin{aligned} \langle \nabla_\theta \mathcal{L}(x) \rangle_{x\sim p_\text{data}} &= \Big\langle -\nabla_\theta \log p_\theta(x) \Big\rangle_{x\sim p_\text{data}} \qquad (3.33) \\ &= \Big\langle \nabla_\theta E_\theta(x) \Big\rangle_{x\sim p_\text{data}} - \Big\langle \nabla_\theta E_\theta(x) \Big\rangle_{x\sim p_\theta}\ . \qquad (3.34) \end{aligned}$$

The first term samples from the training data, the second from the model. According to the sign of the energy in the loss function, the contribution from the training dataset is referred to as positive energy and the contribution from the model as negative energy. One way to look at the second term is as a normalization which ensures that $\mathcal{L} = 0$ for $p_\theta(x) = p_{\text{data}}(x)$. Another way is to view it as inducing a structure for the minimization of the likelihood.

One practical way of sampling from $p_\theta(x)$ is to use Markov-chain Monte Carlo (MCMC). We use Langevin Markov chains (LMC), where the steps are defined by drifting a random walk towards high probability points according to

$$x_{t+1} = x_t + \lambda_x \nabla_x \log p_\theta(x) + \sigma_x \epsilon_t \qquad \text{with} \qquad \epsilon_t \sim \mathcal{N}_{0,1} \; . \tag{3.35}$$

Here, λ is the step size and σ the noise standard deviation. When $2\lambda = \sigma^2$ the equation resembles Brownian motion and gives exact samples from $p_\theta(x)$ in the limit of $t \to +\infty$ and $\sigma \to 0$.

For ML applications, the high dimensionality of the data makes it difficult to cover the entire physics space x with Markov chains of reasonable length. For this reason, it is common to use shorter chains and to choose λ and σ to place more weight on the gradient term than on the noise term. If $2\lambda \neq \sigma^2$, this is equivalent to sampling from the distribution at a different temperature defined as

$$T = \frac{\sigma^2}{2\lambda} \; , \tag{3.36}$$

where σ and λ have been previously defined in Eq. (3.35). By upweighting λ or downweighting σ we are effectively sampling from the distribution at a low temperature, thereby converging more quickly to the modes of the distribution.

By inspecting the expectation value of the loss in the form of Eq. (3.34), we can identify the training as a minmax problem, where we minimize the energy of the training samples and maximize the energy of the MCMC samples. This means that the energy of training data points is pushed downwards. At the same time the energy of Markov chains sampled from the energy model distribution will be pushed upwards. For instance, if $p_\theta(x)$ reproduces $p_{\text{data}}(x)$ over most of the phase space x, but $p_\theta(x)$ includes an additional mode, its phase space region will be assigned large values of $E_\theta(x)$ through the minimization of the loss. This way, all modes present in the energy model distribution but missing from the training distribution p_{data} will be suppressed. This process of adjusting the energy continues until the model reaches the equilibrium in which the model distribution is identical to the training data distribution.

Despite the well-defined algorithm, training EBMs is difficult due to instabilities arising from (i) the minmax optimization, with similar dynamics to balancing a generator and discriminator in a generative adversarial network; (ii) potentially biased sampling from the MCMC due to a low effective temperature; and (iii) instabilities in the LMC chains. Altogether, stabilizing the training during its different phases requires serious effort.

3.2.3 Representation learning

The final part concerns the definition of representations suitable for either classification or generation purposes. A practical way of approaching the problem is the definition of pre-processing steps hand-crafted to the specific data and often physically not well-motivated. The alternative approach consist of defining a pseudo-task which we use as training objective. For a bijective transformation, a new representation can be seen intuitively as a variable transformation following again Eq. (3.24). This means that a proper choice of the Jacobian can ease the learning process of the target density.

The two examples relevant for this thesis are concerned with high-dimensional data. In the first approach we assume that the data lives on a much smaller manifold which we learn in an completely unsupervised way. The second example instead learns approximate known symmetries in the data in a self-supervised approach.

Autoencoders

AEs are the simplest tools for unsupervised representation learning. It is defined as two-module function that maps an input to its reconstruction using an encoder-decoder structure,

$$f_\theta(x): \quad \mathbb{R}^{d_x} \longrightarrow \mathbb{R}^{d_z} \tag{3.37}$$

$$g_\phi(z): \quad \mathbb{R}^{d_z} \longrightarrow \mathbb{R}^{d_x}\ , \tag{3.38}$$

where the dimensionality of the latent space d_z is smaller than input space d_x. The encoder f_θ and the decoder g_ϕ are two neural networks with generally uncoupled parameters θ and ϕ. The training minimizes the difference between the original input and its mapping. A typical choice for the loss function is the MSE of this reconstruction, as defined in Eq. (3.6). While minimizing the reconstruction error, the network encodes the important information in the latent space. For high-dimensional datasets it can be the real manifold of the data, if the latent space is large enough, or the most prominent features of the training data in case of a very small latent space.

One of the original application of autoencoders in particle physics is anomaly detection [139]. An autoencoder trained on a background dominated sample encodes information only relevant to its reconstruction. This means that OOD samples are likely to have large reconstruction error and are, therefore, in the tail of the MSE distribution. Following this observation, the reconstruction error $s(x)$ can be used as an anomaly score which defines the OOD region,

$$\text{OOD}: \Big\{ x \in \mathbb{R}^{d_x} | s(x) > \tau \Big\}\ , \tag{3.39}$$

with a sensitive choice of the threshold τ. However, any transformation of the background distribution can change the definition of anomalies [100] and a motivated transformation is presented in Ch. 5.

This definition a vanilla AE is not a probabilistic model, since a small reconstruction error does not correspond to a large likelihood. A common way to promote the AE to a probabilistic model is the introduction of a latent prior distribution and a likelihood assumption for the decoder, like in the variational autoencoder. In a VAE we introduce a latent variable z and we assume to describe the true posterior distribution $p_\theta(z|x)$

with a variational approximation $q_\phi(z|x)$, i.e. the encoder. If we try to minimize the KL-divergence between the two distribution, we obtain

$$\mathrm{KL}(q_\phi(z|x)|p_\theta(z|x)) = -\big\langle \log(p_\theta(x)) \big\rangle_{q_\phi(z|x)} + \Big\langle \log \frac{p_\theta(x,z)}{q_\phi(z|x)} \Big\rangle_{q_\phi(z|x)} . \tag{3.40}$$

Even if the NLL is intractable, we can minimize the latter term. In fact the positivity of the KL-divergence, ensures that this is a lower bound of the NLL and therefore called the Evidence Lower Bound (ELBO) loss,

$$\begin{aligned} \mathcal{L}_\mathrm{ELBO} &= \Big\langle \log \frac{p_\theta(x,z)}{q_\phi(z|x)} \Big\rangle_{q_\phi(z|x)} & (3.41)\\ &= \big\langle \log p_\theta(x|z) \big\rangle_{q_\phi(z|x)} + \mathrm{KL}(q_\phi(z|x)|p_\theta(z)) & (3.42) \end{aligned}$$

The first term minimizes the conditional likelihood of the decoder, while the second is a regularization term that pushes the latent distribution to a prior distribution. If we assume a Gaussian latent prior with fixed diagonal standard deviation and we add a weight β between the two components of the loss, we obtain the β-VAE [140]. Although this can be seen as a generative model, the quality of the samples is nowadays far from more modern architectures. Therefore, we will use the β-VAE as a regularization term to learn a smooth and compact latent representation of high-dimensional sparse data. An alternative likelihood-based approach that reduces the latent space assumptions inherits from energy-based networks and is discussed in Ch. 5.

Contrastive learning

The contrastive learning (CLR) of visual representations [101] framework showed how to learn representations imposing approximate invariance under specified transformations. The alternative approach directly enforces the invariance in the network. For instance, many networks in particle physics revolve around Lorentz invariance or equivariance [126, 141, 142]. However, this is not always possible and a more general approach augments the data to learn an approximate symmetry during training. CLR follows the latter by introducing these elements:

- a latent space z, the representation space;
- a set of transformations (or augmentations) we want to apply to the training data. We want to be invariant under these tranformations;
- pseudo-labels used during training to specify if two points should be mapped to the same representation;
- a compact distance measure in the latent space;
- a loss function that forces invariance and uniformity of the representations.

This time we only need an encoder neural network f,

$$f_\theta(x): \quad \mathbb{R}^{d_x} \longrightarrow \mathbb{R}^{d_z} , \tag{3.43}$$

where the dimensionality of the representation space can now also be larger than the original space. The pseudo-labels we define during training are positive and negative pairs,

$$\begin{aligned}
&\text{positive pairs:} \qquad && \{(x_i, x'_i)\}_{i=1}^{N} \\
&\text{negative pairs:} \qquad && \Big\{(x_i, x_j) \cup (x_i, x'_j)\Big\}_{i,j=1,i\neq j}^{N} \,,
\end{aligned} \tag{3.44}$$

where the primed objects are transformed with the set of augmentations. The positive pair only considers the object with its augmented version. Instead, the negative pairs relate two different objects with and without the application of the augmentations. The distance measure we use for each pair is the cosine similarity, defined as

$$s(z_i, z_j) = \frac{\cos\theta_{ij}}{\|z_i\|\|z_j\|} \,, \tag{3.45}$$

where θ_{ij} is the angle between the two vectors in the representation space. The cosine similarity measures the distance between points in a compact latent space $z \in \mathbb{S}^{d_z-1}$. On the hyper-sphere maximizing the distance between pairs leads to a uniform distribution, i.e. it is not possible to separate two representation by increasing the norm of the vector.

Finally, the CLR loss function is designed to minimize the distance between positive pairs and maximize it for negative pairs, after mapping them to the representation space. These two terms, also called alignment and uniformity, are defined as

$$\mathcal{L}_{\text{align}} = \langle s(z_i, z'_i)/\tau \rangle_{p_{\text{data}}} \tag{3.46}$$

$$\mathcal{L}_{\text{unif}} = \langle \log \sum_{j\neq i} e^{(-s(z_i,z_j)/\tau)} + e^{(-s(z_i,z'_j)/\tau)} \rangle_{p_{\text{data}}} \,, \tag{3.47}$$

where τ is a temperature hyper-parameter. Combining the two terms in a single loss, we obtain the CLR loss function

$$\begin{aligned}
\mathcal{L}_{\text{CLR}} &= -\mathcal{L}_{\text{align}} + \mathcal{L}_{\text{unif}} \\
&= -\log \frac{\exp(s(z_i, z'_i)/\tau)}{\sum_{j\neq i} \exp(s(z_i, z_j)/\tau) + \exp(s(z_i, z'_j)/\tau)} \,,
\end{aligned} \tag{3.48}$$

Notice that the numerator is minimized if $s(z_i, z'_i) = 1$, i.e. the representations of the two points z_i and z'_i are identical on the hyper-sphere. To keep the representation informative, this is coupled with the uniformity loss. The final solution will be approximately invariant under the applied transformations. We extend this framework to anomaly detection in Ch. 5.

3.3 Architectures

Each model presented in the previous section can be implemented in multiple ways, imposing different inductive biases on the learnable family of functions. In this section, we describe the details of the architectures used in the later chapters.

3.3.1 Normalizing flows

The naive implementation of a normalizing flow, using the loss function in Eq. (3.25), has several limitations we have to account for. The first problem is the scalability of the computation of the Jacobian of the transformation. A complete general flow architecture has a $\mathcal{O}(N^3)$ computational scaling which makes the training unfeasible for high-dimensional problems. This is the case for our applications and we present a design choice that solve this issue. Secondly, we still have to specify a fully invertible neural network needed for the bijective mapping.

Affine coupling blocks We reduce the computational cost of the Jacobian with coupling blocks. In a coupling block the D dimensional input vector x is transformed as:

$$\begin{cases} y_i = x_i \quad i \in 1, \ldots, d \\ y_i = f_\theta(x_i | x_1, \ldots, x_d) \quad i \in d+1, \ldots, D, \end{cases} \tag{3.49}$$

where $d = D/2$ is a standard splitting choice. The peculiar property of Eq. (3.49) is its expression of the Jacobian,

$$\frac{\partial y}{\partial x} = \begin{pmatrix} \mathbb{I} & (\neq 0) \\ 0 & \frac{\partial y_i}{\partial x_i} \end{pmatrix} \tag{3.50}$$

which is a triangular matrix with determinant given by the product of the diagonal elements. This transformation introduces limitations on the expressivity of the network but it also reduces the scaling to a $\mathcal{O}(N)$ operations. We will refer to this networks as invertible neural networks (INN) [143] or coupling block networks [144].

It is left to define an invertible transformation. The affine transformation

$$y_i = x_i s_{\theta i} + t_{\theta i} \tag{3.51}$$

fulfills this role. Indeed, given the features y, it is possible to retrieve the original inputs x. The parameters s, t are predicted by a neural network which can be a simple, non-invertible, stack of linear layers with non-linearities. A clear issue, which stems from this formulation, is that a subset of the features is unchanged. This is solved by stacking a fixed number T of coupling blocks, each one followed by a permutation of the input vector. The original loss function is therefore modified to

$$\begin{aligned} \mathcal{L}_{\text{INN}} &= \langle -\log p_\theta(x) \rangle_{p_{\text{data}}} && (3.52) \\ &= \langle -\log p_{\text{lat}}(g_T \circ \ldots \circ g_0(x)) \rangle_{p_{\text{data}}} + \Big\langle \sum_i^T \log \left| \frac{\partial g_i}{\partial x} \right| \Big\rangle_{p_{\text{data}}}, && (3.53) \end{aligned}$$

where we recognize $\bar{G}(x) = g_T \circ \ldots \circ g_0$, now a composition of the T transformations $g_0, \ldots, g_T$. Additionally, a deeper network can also improve the density estimate of the phase space point by sequentially transforming the distribution to the desired latent space instead of having one single step. Among the permutations used between the blocks, a random permutation defined at the initialization stage is a standard choice. Other possibilities are a flip of the two subvectors [144], rotations [143], or learnable permutations [145].

An alternative approach is the masked autoregressive flow (MAF) [146], which predicts the output vector according to

$$p(x,t) = p(x_0)p(x_1|x_0)\dots p(x_D|x_0,\dots,x_{D-1}) \tag{3.54}$$

However, this approach has a slow sampling process due to the additional number of network evaluations needed in the inverse pass. Proposed alternatives have fast sampling but slow training, the inverse autoregressive flow (IAF) [147], or require an additional teacher-student distillation process where a fast student network is trained to ad-hoc reproduce the teacher density estimate [23].

Neural spline flows A considerably more expressive invertible transformation for a coupling block normalizing flow is a spline parameterization. Assuming that we want to define a spline in the interval $[0,1]$, a neural network predicts the heights, widths, and derivatives of K predetermined bins. These are processed with a Softmax function to ensure the correct normalization. From the $[0,1]$ window it is possible to rescale the function to cover a different interval $[a,b]$. Outside the spline we apply a linear function. The two spline transformations we will use in Ch. 4 are rational quadratic splines [148], and cubic splines [149]. The rational quadratic spline fits in each bin a function

$$f_k(\xi) = \frac{\alpha_{k0}\xi^2 + \alpha_{k1}\xi(1-\xi) + \alpha_{k2}\xi^2}{\beta_{k0}\xi^2 + \beta_{k1}\xi(1-\xi) + \beta_{k2}\xi^2}\,, \tag{3.55}$$

while the cubic spline fits the polynomial of order three

$$g_k(\zeta) = \delta_{k0} + \delta_{k1}\zeta + \delta_{k2}\zeta^2 + \delta_{k3}\zeta^3\,. \tag{3.56}$$

The detail of the parameterizations, including the calculation of the inverse and the Jacobians, can be found in the respective references.

3.3.2 Transformers

The key aspect of transformers is the attention mechanism. This operation builds weights based on the importance between a query and a key vector. These weights are then used to transform the input similarly to the change of basis formula. Mathematically, we construct the self-attention in which the queries and vectors are all obtained from the input vector. The prediction of the prefactors a_i depends on learnable weights according to

$$\begin{aligned} x_i' &= \sum_{j=1}^{N_c} a_j v_j \\ &= \sum_{j=1}^{N_c} \text{Softmax}_j \left(\frac{(W^Q x_i)\cdot(W^K x_j)}{\sqrt{d_z}} \right) W^V x_j\,, \end{aligned} \tag{3.57}$$

where W^Q and W^K are learnable matrices used to construct the query $W^Q x$ and the key $W^K x$, and d_z is a normalization factor equal to the dimensionality of the query x_i. The matrix W^V is also a learnable matrix used to define the value vector $W^V x$. The Softmax operation ensures that the a_j are a set of properly normalized weights, also called attention weights, which are applied to the value vector v_j. This is often referred to as a single-head self-attention. The multi-head operation simply splits the self-attention into separate learnable weight matrices over the input features dimension.

3.3.3 Conditional Flow Matching

A limitation of the normalizing flow architecture is the finite number of transformations we can apply to the base distribution. A conditional flow matching (CFM) model can be seen as an extension of a normalizing flow with a continuous evolution encoded in the time variable t. It starts with the ordinary differential equation (ODE)

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v(x(t), t) \qquad \text{with} \qquad x \in \mathbb{R}^d \, , \tag{3.58}$$

and a velocity field $v(x(t), t) \in \mathbb{R}^d$. This time evolution is related to the underlying density through the continuity equation

$$\frac{\partial p(x,t)}{\partial t} + \nabla_x \left[p(x,t) v(x,t) \right] = 0 \, . \tag{3.59}$$

Our aim is to find a velocity field that transforms the density $p(x, t)$ such that

$$p(x,t) \to \begin{cases} \mathcal{N}(x; 0, 1) & t \to 1 \\ p_{\text{data}}(x) & t \to 0 \, . \end{cases} \tag{3.60}$$

We estimate the velocity field with $v_\phi(x(t), t)$, a neural network. For our purposes, we can sample the data distribution from Gaussian random numbers, tracing the trajectory using any ODE solver. We can train this network with a simple MSE loss,

$$\mathcal{L} = ||v(x,t) - v_\phi(x,t)||_{L_2} \tag{3.61}$$

Defining the training trajectories to be linear, the conditional path for an initial point x_0 is

$$x(t|x_0) = (1-t)x_0 + t\epsilon \qquad \epsilon \sim \mathcal{N}(0,1) \tag{3.62}$$

Then, we can define the velocity field $v(x, t)$ as [129]

$$v(x,t) = \int \mathrm{d}x_0 \frac{v(x,t|x_0)\, p(x,t|x_0)\, p_{\text{data}}(x))}{p(x,t)} \, , \tag{3.63}$$

where $p(x, t|x_0) = \mathcal{N}(x; (1-t)x_0, t)$. With this definition, we write the final loss function, where we have to sample from the training data, the time $t \in \mathcal{U}(0, 1)$ and the latent noise $\epsilon \sim \mathcal{N}(0, 1)$:

$$\mathcal{L}_{\text{CFM}} = \left\langle \left[v_\phi((1-t)x_0 + t\epsilon, t) - \frac{\mathrm{d}x(t|x_0)}{\mathrm{d}t} \right]^2 \right\rangle_{U(0,1), \mathcal{N}, p_{\text{data}}} \tag{3.64}$$

$$= \left\langle \left[v_\phi((1-t)x_0 + t\epsilon, t) - (\epsilon - x_0) \right]^2 \right\rangle_{U(0,1), \mathcal{N}, p_{\text{data}}} \, . \tag{3.65}$$

Similarly to the conditional normalizing flow, conditional probability distributions can be learned by allowing v_ϕ to depend on additional inputs.

After training, the sampling procedure requires solving the ODE from $t = 1$ to $t = 0$

using the learned velocity field v_ϕ,

$$x(t=0) = x(t=1) - \int_0^1 v_\phi(x,t)\mathrm{d}x\,. \tag{3.66}$$

CHAPTER 4

Fast detector simulations

The content of this chapter was finalized in collaboration with Ranit Das, Florian Ernst, Theo Heimel, Claudius Krause, Ayodele Ore, Sofia Palacios Schweitzer, Tilman Plehn, and David Shih.

In the development of the LHC as a precision-hadron collider, the detector simulation has become a major bottleneck in speed and precision, in particular the reproduction of the detailed interactions of incident and secondary particles within the calorimeters. Generating these calorimeter showers with GEANT4 [118,119,150], based on first principles, takes a substantial amount of the LHC computing budget. Without significant progress, simulations will be the limiting factor for all analyses at the high-luminosity upgrade of the LHC.

In this chapter, we will focus on the problem of building fast and accurate surrogate models for calorimeter shower simulation using cutting-edge generative networks. We embark in the challenge of simulating the detector response in calorimeters with increasing granularity. We will cover both aspects of the trade-off between generation speed and quality.We will first push the sampling speed frontier in Sec. 4.3 with a normalizing flow network. Then, we maximize the faithfulness of the showers in Sec. 4.4 with a transformer-based CFM model at the expense of slower sampling speed.

E_{inc}	256 MeV ... 131 GeV	256 GeV	0.512 GeV	1.04 TeV	2.1 TeV	4.2 TeV
photons	10000 per energy	10000	5000	3000	2000	1000
pions	10000 per energy	9800	5000	3000	2000	1000

Table 4.1: Sample sizes for different incident energies in dataset 1.

4.1 High-dimensional calorimeters

The showers we aim to generate are represented as energy deposition in a fixed spatial grid. Therefore, a shower consists of a vector containing the energy deposition in each voxel of the 3-dimensional geometry. The grid can be different from the real detector and in a second step, which we do not address here, the shower has to be mapped from the voxelized space back to detector space.

We use the public datasets [151–154] of the Fast Calorimeter Simulation Challenge [155]. They consist of showers simulated with GEANT4 for different incident particles. The general geometry is the same across all datasets: the detector volume is segmented into layers in the direction of the incoming particle. Each layer is segmented along polar coordinates in radial (r) and angular (α) bins. A shower is given as the incident energy of the incoming particle and the energy depositions in each voxel.

Dataset 1 (DS1) provides calorimeter showers for central photons and charged pions. They have been used in ATLFAST3 [22]. The voxelizations for photons and pions in the radial and angular bins ($n_r \times n_\alpha$) are

$$\begin{aligned} \text{photons} \quad & 8\times 1,\ 16\times 10,\ 19\times 10,\ 5\times 1,\ 5\times 1 \\ \text{pions} \quad & 8\times 1,\ 10\times 10,\ 10\times 10,\ 5\times 1,\ 15\times 10,\ 16\times 10,\ 10\times 1 \end{aligned} \tag{4.1}$$

This gives 368 voxels for photons and 533 voxels for pions. The incoming particles are simulated for 15 different incident energies $E_{\text{inc}} = 256$ MeV ... 4.2 TeV, increasing by factors of two, with the sample sizes given in Tab. 4.1. The original ATLAS dataset does not require an energy threshold. The effect of a threshold on the shower distributions at the detector cell level requires further studies. We require $E_{\text{min}} = 1$ MeV to all generated voxels, motivated by the readout threshold of the calorimeter cells and the fact that photon showers require a minimum cell energy of 10 MeV to cluster and pion showers start clustering at 300 MeV [156]. We show an example shower from dataset-1 in Fig. 4.1.

Datasets 2 and 3 (DS2/3) are not modeled after existing detectors. They assume 45 layers of active silicon detector (thickness 0.3 mm), alternating with inactive tungsten absorber layers (thickness 1.4 mm) at $\eta = 0$. Each dataset contains 100,000 GEANT4 positron showers with log-uniform $E_{\text{inc}} = 1$ ... 1000 GeV. The only difference between the two datasets is the voxelization. In dataset 2, each layer is divided into 16×9 angular and radial voxels, defining 6480 voxels in total. Dataset 3 uses 50×18 voxels per layer or 40,500 voxels in total. The minimal recorded energy per voxel for these two datasets is 15.15 keV.

Clever preprocessing We improve our training by including a series of preprocessing steps, similar to previous studies [5, 29, 31, 37, 38]. We split information on the deposited energy from its distribution over voxels by introducing energy ratios [21]

$$u_0 = \frac{\sum_i E_i}{f E_{\text{inc}}} \qquad \text{and} \qquad u_i = \frac{E_i}{\sum_{j\geq i} E_j} \quad i = 1, \dots, 44\ , \tag{4.2}$$

where E_i refers to the total energy deposited in layer i, and $f \in \mathbb{R}$ is a scale factor. The number of u-variables matches the number of layers. With these variables extracted from a given shower, we are free to normalize the voxel values by the energy of their corresponding layer without losing any information. This definition is analytically invertible, imposes energy conservation, and ensures that the normalized voxels and each $u_{i>0}$ are always in the range $[0, 1]$. However, due to the calibration of the detector response caused by the inactive material, u_0 can have values larger than 1. We set $f = 2.85$ in Eq. (4.2), to rescale $u_0 \in [0, 1]$. All networks are conditioned on E_inc. This quantity is passed to the network after a log transformation and a rescaling into the unit interval.

To train the autoencoders used for dimensionality reduction we do not use any additional preprocessing steps. For the setup using the full input space, we apply a logit transformation regularized by the parameter α which rescales each input voxel x,

$$\begin{aligned} x_\alpha &= (1-2\alpha)x + \alpha \in [\alpha, 1-\alpha] \qquad \text{with} \quad \alpha = 10^{-6} \\ x' &= \log \frac{x_\alpha}{1-x_\alpha} \, . \end{aligned} \tag{4.3}$$

Finally, we calculate the mean and the standard deviation of the training dataset and standardize each feature. The postprocessing includes an additional step that rescales the sum of the generated voxels to ensure the correct normalization in each layer.

The final task can be formulated in two ways:

- The u_i are appended to the list of voxels for each shower and a single network is trained on this enlarged vector. We train the INN in this way, conditioned on the logarithm of the incident energies. Unlike, for instance, CaloFlow [23, 29, 36] we train a single network without any distillation, maximizing training and generation speed;
- The set of u_i is generated separately by an "energy" network. The energy network learns the energy-ratio features conditioned on the incident energy, $p(u_i| \, E_\text{inc})$. This approach provides a better modeling of the energy deposition in each layer due to the smaller space tackled by the generative network. This will be the strategy in Sec. 4.4.

4.2 Evaluation of the generated showers

Evaluating the showers generated by the networks is a challenging problem due to the high-dimensionality of the space of interest. We start by exploring several physics motivated high-level observables. The specific physical features of a shower are governed

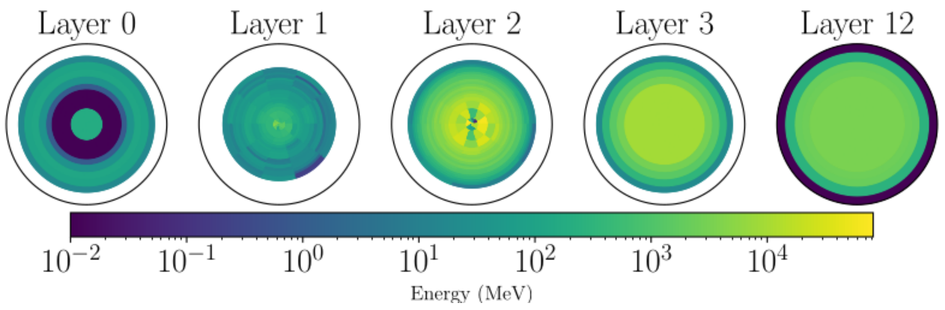


Figure 4.1: Example γ shower from DS1. The layers in the depth direction are separated in slices. The minimum value on the energy scale corresponds to no energy deposition.

by its incident energy. Low-energy showers will interact with only a few layers of the calorimeter and quickly widen, leading to a broad center of energy distribution in earlier calorimeter layers and a high sparsity in the given voxelization. High-energy showers penetrate the calorimeter more deeply. They will be collimated in the initial layers and have low sparsity since each shower is likely to deposit energy in each voxel.

To see if the ML-learned showers reflect these physics properties, we look at physics-motivated and high-level features. Given a shower with energy depositions $\mathcal{I}$, we look at the center of energy and its width for each layer,

$$\langle\zeta\rangle = \frac{\zeta \cdot \mathcal{I}}{\sum_i \mathcal{I}_i} \qquad \text{and} \qquad \sigma_{\langle\zeta\rangle} = \sqrt{\frac{\boldsymbol{\zeta^2} \cdot \mathcal{I}}{\sum_i \mathcal{I}_i} - \langle\zeta\rangle^2} \qquad \text{for} \qquad \zeta = \eta, \phi \; ; \tag{4.4}$$

where $\sum_i$ runs over the voxels in one layer. We also look at the energy deposition in each layer; the layer sparsity; and for DS1, the ratio $E_\text{tot}/E_\text{inc}$ for each discrete incident energy.

In addition to the layer-wise shower shapes, we calculate the mean shower depth weighted by the energy deposition in each of the N layers for slices in the radial direction for DS2 and DS3,

$$d_{r_j} = \frac{\sum_i^N k_i E_{i,r_j}}{E_{\text{tot},j}} \qquad\qquad r_j \in \{0, \dots, |r|\} \; . \tag{4.5}$$

Here E_{i,r_j} is the average energy deposition in slice r_j, and $E_{\text{tot},j}$ is the total energy deposition in the selected slice. Slices in the angular direction are less interesting to calculate due to the rotational invariance of the showers.

All these observables are extremely useful for the detection of failure modes in the network, in particular related to important physical quantities. However, they are limited to one-dimensional histograms which do not include correlations. Given a generative model trained on some reference data, we would like to know how well it reproduces the data in the full phase space. This includes correct reproduction of critical high-level features as well as the multi-dimensional correlations between all the features throughout phase space, which might not be visible at the level of histograms of pre-defined high-level features. An optimal binary classifier, trained to distinguish generated from reference data in the full phase space, fits the bill in every respect. By the Neyman-Pearson (NP) lemma, this classifier is the most powerful discriminant between generative model and reference data.

Studies that have used the classifier metric to judge the quality of generative models have tended to focus exclusively on single numbers [21, 23, 27, 29, 30, 36, 157], like the area-under-the-curve (AUC), the loss, or the accuracy of the classifier. While these aggregate measures certainly have their uses, there is much more useful information to be gleaned from the classifier than a single number [31, 43, 158]. For example, a global integral measure such as the AUC will not detect discrepancies in tails of distributions. Also, the AUC becomes less and less informative the closer the generated and reference samples become. Finally, declaring the model with the highest AUC as the “best” model is oversimplistic, because the definition of the “best” generative network depends on what we actually require from the generative network and how we want to use its output.

We choose to work in terms of *weights* which can be obtained from the classifier outputs C as

$$w(x) = \frac{p_\text{data}(x)}{p_\theta(x)} = \frac{C(x)}{1 - C(x)} \qquad \text{with} \qquad C(x) = \frac{p_\text{data}(x)}{p_\text{data}(x) + p_\theta(x)} \; . \tag{4.6}$$

The assumption is that the NP-classifier learns the density ratio. For a good generative model and an optimal classifier, the weight distribution will typically peak near one, with tails to the left ($w \ll 1$) and right ($w \gg 1$), corresponding to regions of phase space where the generative model is overproducing and underproducing the reference data, respectively. On general grounds, the NP classifier should have an excess of generated events as a small-weights tail of the distribution, and an excess of reference events as a large-weight tail. Indeed this is a general pattern we will observe in the different examples we consider.

For these weights it is crucial that we evaluate them on the reference data and on the generated datasets combined, because typical failure modes correspond to tails for one of the two datasets [2]. By examining the generated and reference data as a function of the cut on the classifier, one can zoom in on the most anomalous regions of phase space, i.e. those that are worst-reproduced by the generative model. This facilitates the interpretability of the classifier metric, which could be further enhanced using recent xAI techniques developed in HEP such as Refs. [159, 160] This topic will be discussed in detail in Sec. 4.5.

4.3 CaloINN

We study two different network architectures. First, we benchmark a standard INN and demonstrate its precision and generation speed especially for low-dimensional phase space (DS1). Second, we embed this INN in a VAE, with the goal of describing DS2 with the same physics content, but a much larger phase space dimensionality.

4.3.1 Neural networks

The normalizing flow architecture follows closely the description of Sec. 3.2. We define a bijective mappings between a (Gaussian) latent space z and the physical phase space x as in Eq. (3.23). The INN variant [143, 161] of normalizing flows is completely symmetric in the two directions. After training the network, $p_\text{data}(x) \sim p_\theta(x)$, we use the INN to sample $p_\theta(x)$ from $p_\text{lat}(r)$ [9]. The building block of our INN architecture is the coupling layer [47–49]. We replace the standard affine layer by a more expressive spline transformation. We use different spline transformations, depending on speed and expressivity. For datasets 1, we employ a rational quadratic spline [148], while for dataset 2 we use a cubic spline [149]. All INN hyperparameters can be found in Appendix B.

The INN is implemented using the FrEIA package [162], and is trained with the likelihood loss of Eq. (3.25). The first term ensures that the latent representation remains Gaussian, while the second term constructs the correct transformation to the phase space distribution. Given the structure of $\overline{G}_\theta(x)$ and the latent distribution p_lat, both terms can be computed efficiently.

The problem with the INN is the scaling towards larger datasets with its high-dimensional phase space of 40k voxels, as in DS3. To solve this scaling problem we introduce an additional VAE to reduce the dimensionality of the INN mapping. Differently from [30], we do not estimate the dimensionality of the manifold but rather optimize the reconstruction of the VAE while keeping a low-dimensional latent space. The VAE consists of a preprocessing block, an encoder-decoder combination, and a postprocessing block. Both, the decoder and the encoder are conditioned on the incident energies and additional energy variables. Therefore, we compress normalized showers in the latent space and jointly learn the energy and the latent variables with the INN. During generation, the INN samples into the latent space of the VAE, and the VAE decoder translates this information to the shower phase space. We set the latent space to 50 for dataset 1 and dataset 2.

The goal of our β-VAE is to learn to reconstruct the input data. We assume a Gaussian distribution for the encoder network $p_\theta(z|x)$. For a Gaussian encoder the KL-divergence can be computed analytically, and the prefactor is $\beta = 10^{-9}$. For the decoder we use a Bernoulli likelihood, because it outperforms for example its Gaussian counterpart. The Gaussian decoder does not model the shower geometry well, and it under-populates the low-energy regions. The continuous Bernoulli distribution [163] leads to instabilities, as the average energy deposition in the normalized space is close to zero. Therefore, we use a Bernoulli decoder,

$$D(x|\lambda(z)) = \lambda(z)^x (1-\lambda(z))^{1-x} \; , \tag{4.7}$$

defining the combined VAE loss

$$\mathcal{L}_\text{VAE} = \Bigg\langle \Big\langle x \log \lambda + (1-x) \log (1-\lambda) \Big\rangle_{p_\theta(z|x)} + \beta \left[1 + \log \sigma_E^2 - \mu_E^2 - \sigma_E^2 \right] \Bigg\rangle_{p_\text{data}} \; . \tag{4.8}$$

Because the Bernoulli distribution gives a binary probability we use its continuous mean λ as the prediction for the individual voxels.

The remaining differences between the unit-Gauss prior in the latent space and the encoder are mapped by the INN. Applying a 2-step training we first train the VAE and then train the INN given the learned latent space. This means we pass the encoder means and the standard deviations, as well as the energy variables to the INN. The INN is trained as described above, mapping the latent representation of the VAE to a standard Gaussian. As for the full INN, the energy information is encoded following Eq (4.2) and learned by the latent flow. Both encoder and decoder of the VAE are conditioned to these variables.

For the larger datasets 2, we employ a mixture of a convolutional and a fully connected VAE. Our assumption is that the calorimeter layers do not require information from all the other layers, so we can simplify the structure by compressing consecutive layers jointly in a first-step compression. We use an architecture with fully connected sub-blocks, resembling a kernel architecture with a kernel size k (number of jointly encoded calorimeter layers) and a stride s (distance between two neighboring kernel blocks). After this first compression we concatenate these latent sub-spaces and compress them a second time into our final latent space. For the decoding we reverse this two-step structure. The overlapping regions of the fully connected kernel blocks are summed over.

4.3.2 From DS1 to DS2

Dataset 1 photons We start with the photons in dataset 1, the simplest case in terms of dimensionality and of complexity, since photons only undergo a handful interactions in the calorimeter. In this established benchmark normalizing flows are known to excel. We summarize the most interesting high-level features for the Geant4 training data, the INN generator, and the VAE+INN generator in Fig. 4.2.

For instance for the calorimeter layer 2, we first look at the shower shape in rapidity. Dataset 1 is not a symmetric in η and ϕ, because the shower were not generated around $\eta = \phi = 0$. All showers have the same mean width, regardless of the incident energy. This is captured by both networks at the level of 5% to 20%. A failure mode of the INN is the region $\sigma_{\langle\eta\rangle} < 20$ mm. where the network undershoots the training data by up to 30%. A peculiar feature of these distributions is a small peak at zero, which occurs when at most one voxel per layer receives a hit. These cases are better reproduced by the VAE, whereas the INN tends to produce slightly more collimated showers.

The sparsity λ_2 in the same layer is determined by the energy threshold of 1 MeV. The INN matches the truth over the entire λ-range to 10%, while the VAE struggles. In particular, its showers have too many active voxels, leading to the mis-modeled peak close to zero.

Next, we show the energy depositions in layers 0 and 2. Both networks show comparable performance over the entire energy range and are challenged by the sharp increase in energy around 100 MeV. The energy in layer-2 highlights the effect of discrete incident energies. High-energy showers at fixed energy deposit a similar amount of energy creating steps in the histogram.

Finally, the ratio $E_{\text{tot}}/E_{\text{inc}}$ exhibits a small bias in the energy generation for the VAE+INN towards low energies, artifact of the final threshold in the architecture. For

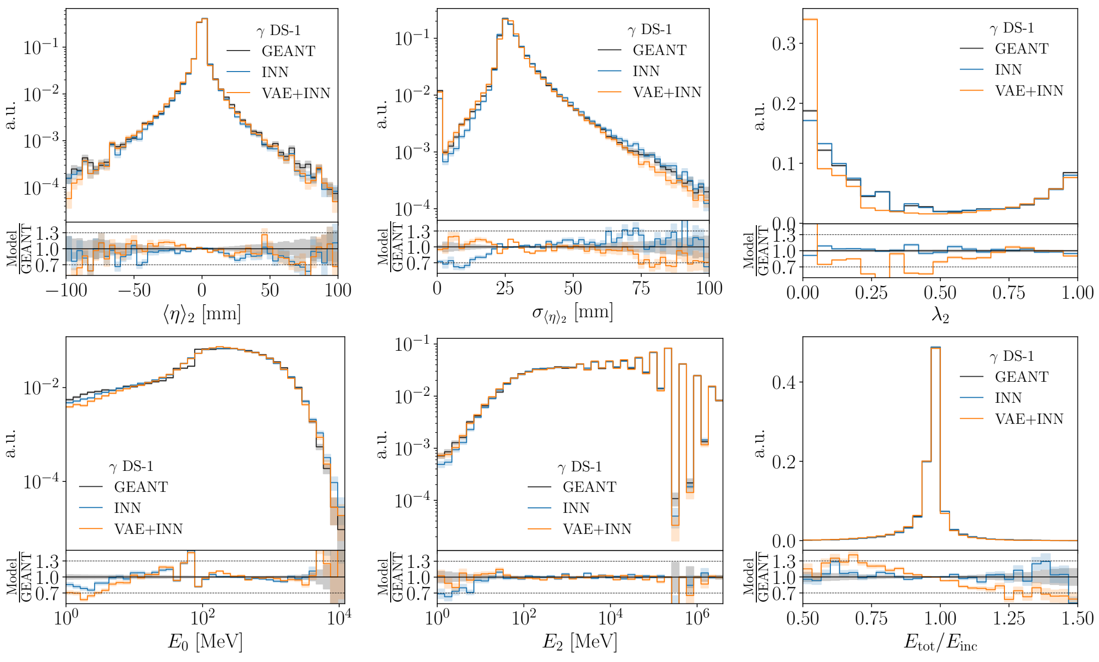


Figure 4.2: Set of high-level features for γ showers in dataset 1, compared between Geant4, INN, and VAE+INN for all incident energies.

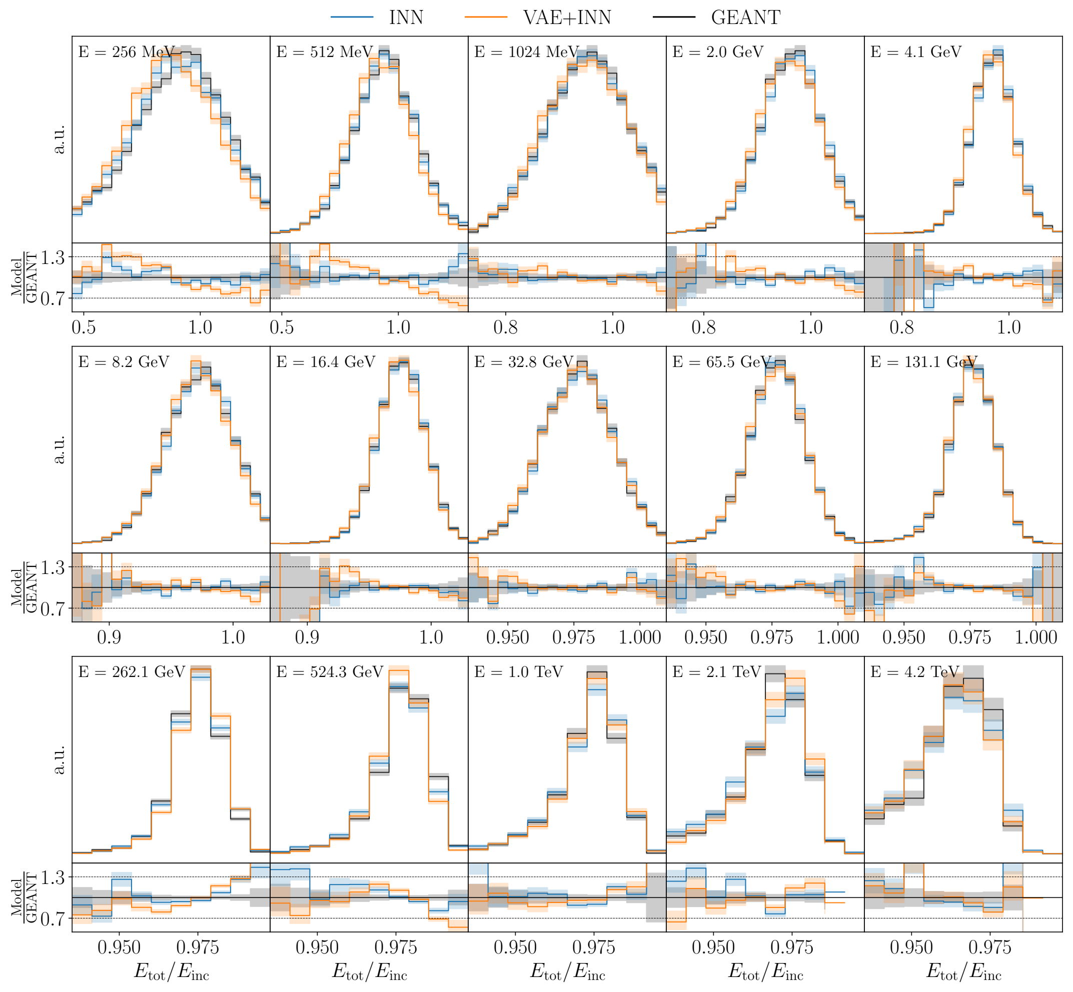


Figure 4.3: Energy ratio $E_{\text{tot}}/E_{\text{inc}}$ for each discrete incident energy, compared between GEANT4, INN, and VAE+INN for γ showers.

smaller incident energies, more voxels are zero [31], which causes a problem for the VAE+INN because we already know that the sparsity is its weakness.

To illustrate the discrete structure of the incident energies in dataset 1, we collect $E_{\text{tot}}/E_{\text{inc}}$ for each incident energy in Fig. 4.3. The incident energy, provided during training and generation, carries energy-dependent information about the shower. For instance, low-energy showers have a much broader energy ratio distribution, in contrast to high-energy showers. Both generative networks learn the conditional distribution on E_{inc} with deviations up to 30% in the tails.

Dataset 1 pions The physics of hadronic showers is significantly more complex than photon showers, so it is interesting to see how our INNs perform for a low-dimensional calorimeter simulation of pions. As before, we show shower shapes, sparsity, energy depositions, and the fraction of deposited energy in Fig. 4.4.

For the shower shapes, both networks show small, percent-level deviations in the bulk of the distributions. In addition, the VAE+INN is smearing out secondary peaks of the distributions. Both networks generate slightly too wide showers, predominantly where the energy deposition ends up in a narrow band of the calorimeter.

The slightly reduced quality of the generated pion showers can also be seen in the sparsity, especially for the VAE+INN. The energy depositions indicate similar failure modes. The sharp cut at low energy is smeared to a different extent by the networks, and a second deviation appears in the low-density region before the sharp cut. From the ratio $E_{\text{tot}}/E_{\text{inc}}$ we see that at all energies the fraction of deposited energy can be very different from shower to shower, leading to the wide energy distribution far from one.

To evaluate the performance of our generative networks on dataset 1 systematically, we train a network to learn the classifier weights defined in Eq.(4.6) over the voxel space. In the left panel of Fig. 4.5 we show the weights for the γ-shower. We clearly see that the INN outperforms the VAE+INN. Its weight distribution peaks much closer to 1 and the corresponding AUC of 0.601 is substantially better than the corresponding AUC=0.936 of the VAE+INN.

More importantly, the INN does not show significant tails at large or small weights, which would indicate distinct failure modes. The peak of the VAE+INN, on the other hand, has moved away from 1. The tail at small weights indicates regions that are overpopulated by the network. We already know that this is the case for the sparsity. Large weights appear in phase space regions which the VAE+INN fails to populate, for instance the widths of the centers of energy.

In the right panel of Fig. 4.5 we see that the two generators perform more similar for π-showers. Both networks now show tails at small and large weights, two orders of magnitude away from one. This means there are regions that are over- and underpopulated by the generative networks. The fact that small weights appear for generated showers and large weights appear for the training data is generally expected. The AUCs for the INN and the VAE+INN are 0.805 and 0.864, respectively. The INN weight distribution is

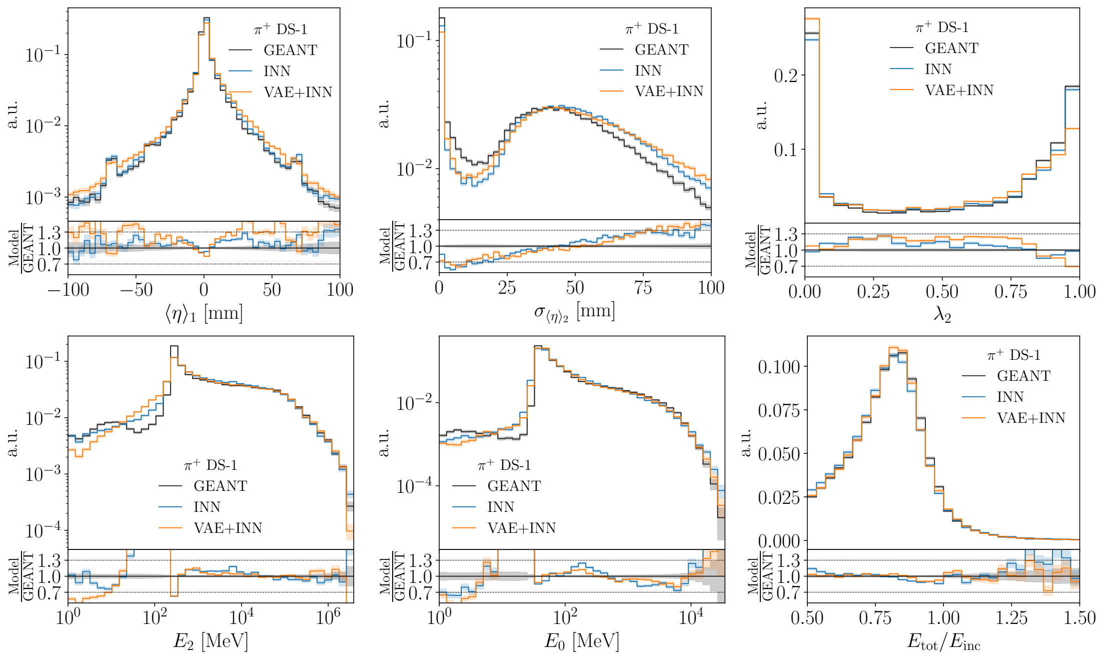


Figure 4.4: Set of high-level features for pion showers in dataset 1, compared between GEANT4, INN, and VAE+INN.

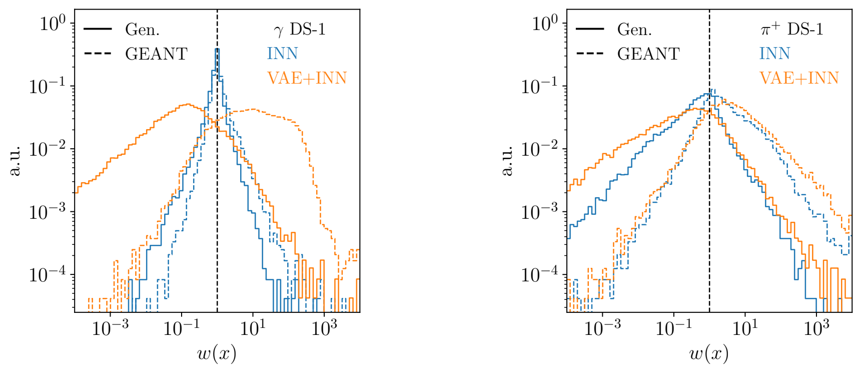


Figure 4.5: Classifier weight distributions in dataset 1. Classifier trained on γ showers (left) and π showers (right).

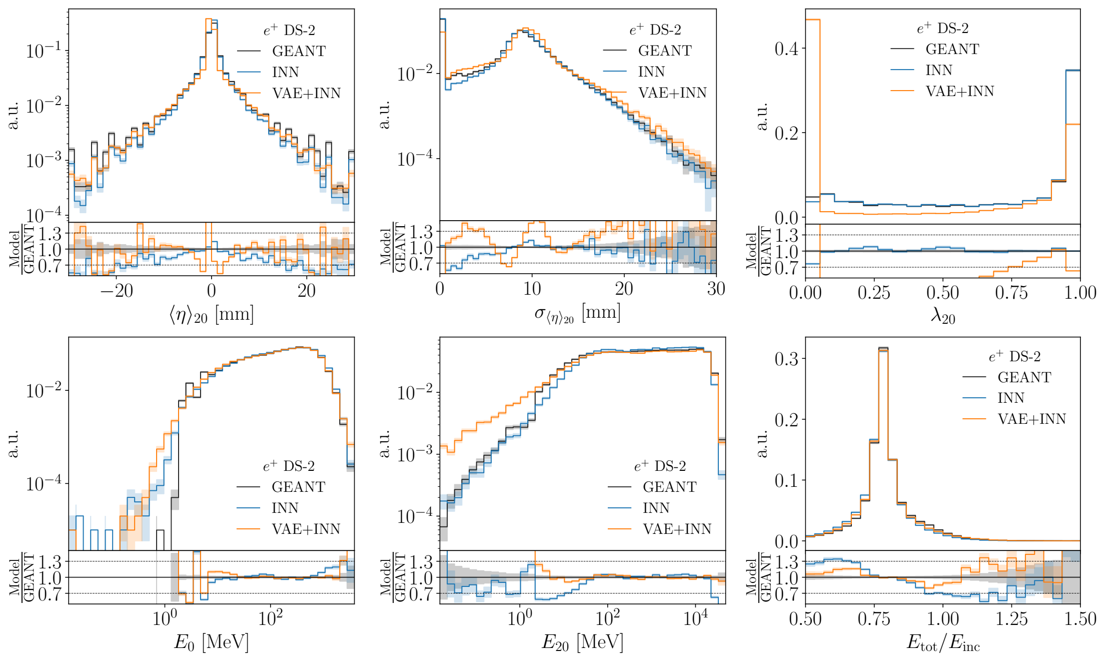


Figure 4.6: Set of high-level features for positron showers in dataset 2, compared between GEANT4, INN, and VAE+INN.

sharper around one, resulting in the smaller AUC compared to the combined VAE+INN approach. Altogether, we find that for dataset 1 with its limited dimensionality of 368 voxels for photons and 533 voxels for pions the INN works well, and that adding a VAE to compress the information does increase with the network performance.

Dataset 2 positrons Dataset 2 is given in terms of 6480 voxels, the kind of dimensionality which will probe the limitations of the regular INN. The number of parameters for this network approaches 200M. The question will be, if the VAE+INN condensation helps the performance of the network. As before, we show a representative set of high-level features in Fig. 4.6. We choose layer 20, approximately in the middle of the calorimeter. It combines features from low-energy showers, which are absorbed in this region, and high-energy showers, which continue until the end of the calorimeter.

From the shower shapes we see that the INN-based architectures generate realistic

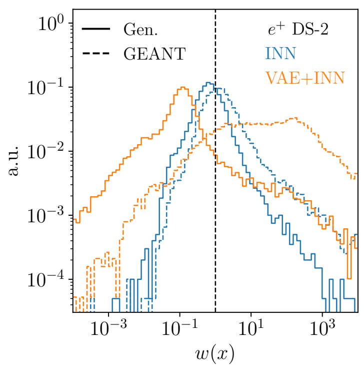


Figure 4.7: Classifier weight distributions. Classifier trained on e^+ showers on dataset 2.

showers at all energies. The training is stable and consistent across different runs of the same architecture. We only see sizeable deviations in the center of energy distributions in the first and last layers of the calorimeter, where there are less energy depositions. The agreement in phase space density between GEANT4 and the INN ranges from a few percent in the bulk of the distributions to 50% in the tails. Similar numbers apply to the width of the center of energy. The failure mode of the INN, regardless of the dataset, is an under-sampling of showers with width between the peak at zero and the secondary peak, for which the location depends on the layer but not on the incident energy. The VAE+INN generates showers of slightly worse quality. This is true for the shower shapes, for instance the peak position in $\sigma_{\langle\eta\rangle}$, but most obvious for the poorly reproduced sparsity.

The two networks learn the energy depositions in the layers in two very different spaces. The INN extracts them with a large number of voxels, while the VAE+INN compresses them into a reduced space of around 50 additional features. This different expressivity is reflected in all energy distributions in Fig. 4.6. While in poorly populated tails the INN does better, for instance at low energies, the VAE+INN performs better for the main features in the central and high-energy regime. This is true for the layer-wise energies, but also for the ratio $E_{\text{tot}}/E_{\text{inc}}$.

Again, we show a systematic comparison for dataset 2 in terms of the classifier weights in the left panel Fig. 4.7. Compared to dataset 1, there is a clear deterioration of the INN performance for the higher-dimensional phase space. At small weights, the tail remains narrow, indicating that there are still no phase space regions where the network over-samples the true phase space distribution. For large weights the weight tail now extends to values larger than $w \sim 10^3$. This tail can be related to a recurrent under-sampling of showers with a small width of the center of energy in each layer, as seen in Fig. 4.6.

The classifier evaluating the VAE+INN generator highlights a few important structures as well. First, we have a clear over-sampled region in phase space with weights $w \sim 10^{-2}$, which we can relate to the center of energy distribution as well. As mentioned before, the VAE+INN over-samples showers with width close to the mean shower width. The classifier confirms this major failure mode. For the large-weight tail we checked that the under-sampled showers do not cluster in the same way, but are distributed over phase space, including tails of distributions.

The AUC values of the classifiers for dataset 2, 0.703 for the INN and 0.916 for the

VAE+INN, confirm the challenge of the INN, especially relative to the well-modelled γ-showers in dataset 1. However, adding a VAE does not significantly improve the situation as long as it is technically possible to train an INN.

Timing comparison Finally, we time our networks using the CaloChallenge procedure. The INN architecture with modern coupling layers is ideally suited for fast and precise generation. We create a singularity container [164] of the software environment and take the time it takes to load the container, load the network, move it on the GPU, generate the samples, and save them to disk. In Tab. 4.2 we show the averaged results from ten runs. We observe a speed–up for increased batch size and when running on the GPU. The INN has a small advantage for dataset 1, but is unable to generate dataset 2 with the highest batch size and dataset 3 altogether. The VAE shows generation times at or below the millisecond mark.

4.4 CaloDREAM

In CaloDREAM, we employ two generative networks, one energy network and one shape network [21]. The energy network learns the energy-ratio features conditioned on the incident energy, $p(u_i|\, E_{\text{inc}})$. The shape network learns the conditional distribution for the voxels, $p(x|\, E_{\text{inc}}, u)$. The two networks are trained independently, but are linked in the generative process. Specifically, to sample showers given an incident energy, we follow

	Batch size	INN		
		1-photon	1-pion	2-positron
GPU	1	24.79 ± 0.49	24.76 ± 0.35	50.90 ± 0.37
	100	0.385 ± 0.002	0.406 ± 0.003	1.900 ± 0.026
	10000	0.162 ± 0.002	0.191 ± 0.006	
CPU	1	17.48 ± 0.09	18.88 ± 0.33	117.5 ± 1.8
	100	0.827 ± 0.028	1.004 ± 0.047	14.26 ± 0.18
	10000	0.510 ± 0.008	0.719 ± 0.016	15.24 ± 1.36
	Batch size	VAE+INN		
		1-photon	1-pion	2
GPU	1	33.64 ± 0.32	33.54 ± 0.23	40.55 ± 0.40
	100	0.507 ± 0.005	0.544 ± 0.007	1.05 ± 0.02
	10000	0.180 ± 0.002	0.228 ± 0.003	0.748 ± 0.018
CPU	1	20.83 ± 0.72	20.05 ± 0.13	28.11 ± 0.15
	100	0.582 ± 0.005	0.886 ± 0.015	1.94 ± 0.01
	10000	0.328 ± 0.004	0.426 ± 0.014	1.25 ± 0.01

Table 4.2: Per-shower generation times in ms. We show mean and standard deviation of 10 independent runs. The star indicates that only 10k samples were generated. The CPU timings were done with an Intel(R) Core(TM) i9-7900X at 3.30 GHz, the GPU timings with an NVIDIA TITAN V with 12GB RAM.

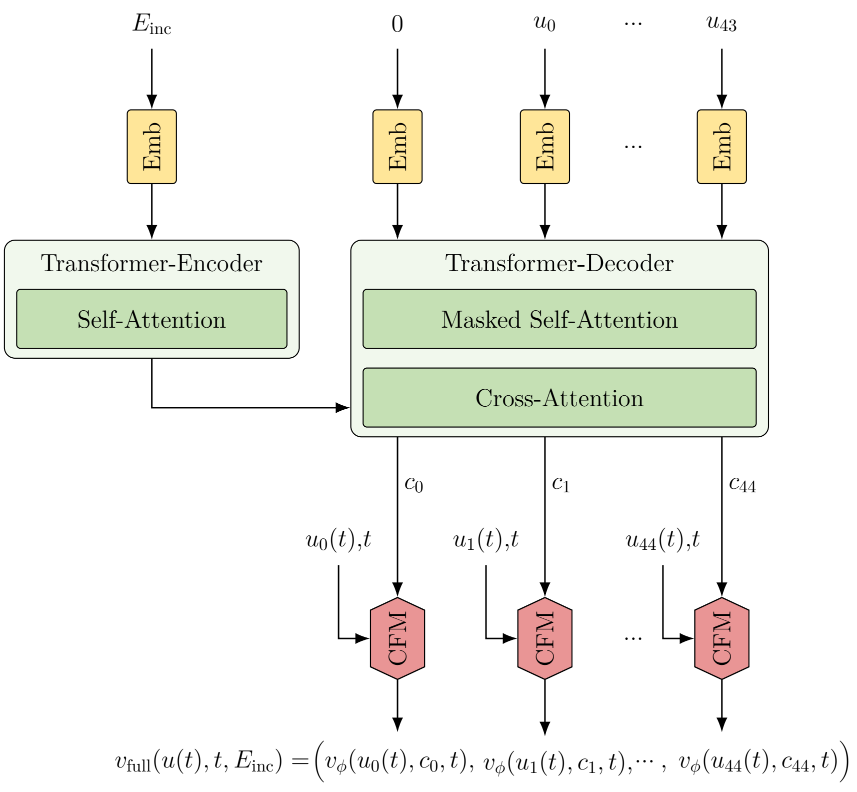


Figure 4.8: Schematic diagram of the autoregressive Transfusion network [166] used in our energy network.

the chain

$$\begin{aligned} u_i &\sim p_\phi(u_i|E_{\text{inc}}) \\ x &\sim p_\theta(x|E_{\text{inc}}, u) \ . \end{aligned} \tag{4.9}$$

In this notation ϕ stands for the weights in the energy network and θ for the weights in the shape network. Although the number of calorimeter layers is consistent across DS2 and DS3 and the underlying showers are the same, we train separate energy networks for each dataset. The incident energy is always sampled from the known distribution in the datasets.

4.4.1 Neural networks

Energy network — Transfusion Both of our generative networks use the conditional flow matching architecture [165]. For the energy network, we exploit the causal nature of the energy deposition in layers using an autoregressive transfusion architecture [166], as visualized in Fig 4.8. We start by embedding E_{inc} as our one-dimensional condition and the u-vector. For the u, this is done by concatenating a one-hot encoded position vector and zero-padding. These embeddings are passed to the encoder and decoder of a transformer, respectively. For the one-dimensional condition the encoder's self-attention reduces to a trivial 1×1 matrix. For the decoder we mask our self-attention with an upper triangle matrix, to keep the autoregressive conditioning. Afterward, we apply a cross-attention between the encoder and decoder outputs. The transformer outputs the

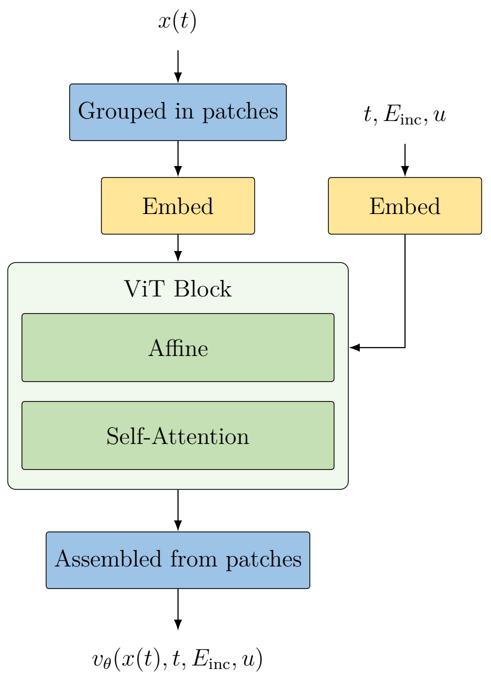


Figure 4.9: Schematic diagram of the vision transformer (ViT) [167] used in our shape network.

vectors $c_0, \dots, c_{44}$, encoding the incident energy and previous energy ratios,

$$c_i = \begin{cases} c_i(u_0, \dots, u_{i-1}, E_\text{inc}) & i > 0 \\ c_i(E_\text{inc}) & i = 0 \ . \end{cases} \tag{4.10}$$

For generation, we use a single dense CFM network v_ϕ, with the inputs time t, embedding c_i, and the point on the diffusion trajectory $u_i(t)$. This network is evaluated 45 times to predict each component of the velocity field individually,

$$v_\text{full}(u(t), t, E_\text{inc}) = (v_\phi(u_0(t), c_0, t), \dots, v_\phi(u_{44}(t), c_{44}, t)) \tag{4.11}$$

During training, we can evaluate the contribution of each u_i to the loss in parallel, whereas sampling requires us to iteratively predict the u_i layer by layer. The hyperparameters of the transfusion network are given in Appendix B.

Shape network — Vision Transformer For the shape network, we use a 3-dimensional vision transformer (ViT) to learn the conditional velocity field $v_\theta(x(t), t, E_\text{inc}, u)$. The architecture is inspired by Ref [167] and illustrated in Fig. 4.9. It divides the calorimeter into non-overlapping groups of voxels, so-called patches, which are embedded using a shared linear layer and passed to a sequence of transformer blocks. Each block consists of a multi-headed self-attention and a dense network that transforms the patch features. To break the permutation symmetry among patches, we add a learnable position encoding to the patch embeddings prior to the first attention block. After the last block, a linear layer projects the processed patch features into the original patch dimensions, where each entry represents a diffusion velocity. Finally, the patches are reassembled into the calorimeter shape.

The network uses a joint embedding for the conditional inputs, t, E_{inc} and u. The time and energy coordinates are embedded with separate dense networks, then summed into a single condition vector. The attention blocks incorporate this condition via affine transformations with shift and scale $a, b \in \mathbb{R}$ and an additional rescaling factor $\gamma \in \mathbb{R}$ learned by dense layers. These are applied within each block, and also to the final projection layer. Concretely, the operation inside the ViT block is summarized by

$$\begin{aligned} x_{\text{h}} &= x + \gamma_{\text{h}} g_{\text{h}}(a_{\text{h}} x + b_{\text{h}}), \\ x_{\text{l}} &= x_{\text{h}} + \gamma_{\text{l}} g_{\text{l}}(a_{\text{l}} x_{\text{h}} + b_{\text{l}}), \end{aligned} \tag{4.12}$$

where g_{h} is the multi-head self-attention step and g_{l} is the fully connected transformation. The hyperparameters of our transformer are given in Appendix B.

The scalability of this architecture is closely tied to the choice of patching. On the one hand, small patches result in high-dimensional attention matrices. While this gives a more expressive network, the large number of operations can become a limitation for highly-granular calorimeters. Conversely, a large patch size compresses many voxels into one object, implying a faster forward pass but at the expense of sample quality. In this case, an expanded embedding dimension is needed to keep the network flexibility fixed.

Usually, we train Bayesian versions [168] of all our generative networks, including calorimeter showers [5]. In this study, the networks learning DS2 and DS3 are so heavy in terms of operations, that an increase by a factor two, to learn an uncertainty map over phase space, surpasses our typical training cost of 40 hours on a cutting-edge NVIDIA H100 GPU. In principle, Bayesian versions of all networks used in this study can be built and used to quantify limitations, for instance related to a lack of training data.

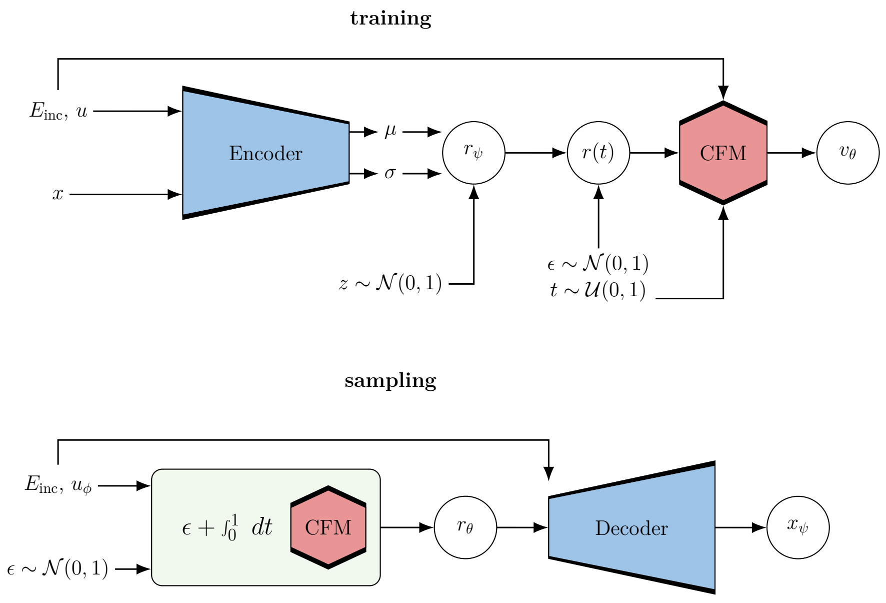


Figure 4.10: Training (upper) and sampling (lower) with the latent diffusion network, using a variational autoencoder.

Latent diffusion As the calorimeter granularity is increased from DS2 to DS3, the computational requirements to train a network on the full voxel space also increase considerably due to the larger number of patches. This motivates a study of how the naive scaling may be avoided by a lower-dimensional latent representation. Starting from the detector geometry, a voxel-based representation of a shower defines a grid with fixed size and stores the deposited energy in each voxel. This means a highly granular voxelization will produce a large fraction of zero voxels, but the showers should define a lower-dimensional manifold of the original phase space. Such a manifold can then be learned by an autoencoder [5, 30, 169].

We train a variational autoencoder with learnable parameters ψ. The encoder outputs a latent parameter pair (μ, σ), which defines the latent variable $r = \mu + z \cdot \sigma$ with $z \sim \mathcal{N}(0,1)$. The encoder distribution represents the phase space distributions over x through $p_\psi(r|x,u)$. For simplicity, in the following we drop the energy dependence in the encoder and decoder distributions. After sampling the latent variable, we minimize the learned likelihood of a Bernoulli decoder $p_\psi(x|r)$ represented by the reconstruction loss

$$\mathcal{L}_\text{VAE} = \big\langle - \log p_\psi(x|r) \big\rangle_{p_\text{data}, p_\psi(r|x)} + \beta \big\langle D_\text{KL}[p_\psi(r|x), \mathcal{N}(0,1)] \big\rangle_{p_\text{data}} \,. \tag{4.13}$$

This choice of likelihood is possible since our preprocessing normalizes voxels into the range $[0,1]$. The reconstruction quality achieved in the autoencoder training places an upper bound on the quality of a generative model trained in the corresponding latent space.

The KL-divergence term, with unit-Gaussian prior and a small weight $\beta = 10^{-6}$, is a regularization rather than a condition for a tractable latent space. It encourages a smooth latent space, over which we train the generative network. Especially for DS3, an autoencoder trained without KL-regularization produces a sparse latent space with features mapped over several orders of magnitude.

The VAE consists of a series of convolutions, the last of which downsamples the data. This structure is mirrored in the decoder using ConvTranspose operations. As always, the energy conditions are encoded in a separate network and passed to the encoder and decoder. For a compressed latent space the ratio between the dimensionality of x and r defines the reduction factor F. Rather than estimating the dimensionality of the datasets, we use a moderate, fixed reduction factor $F \simeq 2.5$ and a bottleneck with two channels. We provide more details on the autoencoder training in Appendix A.

The trained autoencoder is used as a pre- and postprocessing step for the CFM as illustrated in Fig. 4.10. Given the trained encoder distribution $p_\psi(r|x)$ the velocity field $v(r(t),t)$ imposes the boundary conditions

$$p(r,t) \to \begin{cases} \mathcal{N}(r;0,1) & t \to 0 \\ p_\psi(r|x) & t \to 1,\; x \sim p_\text{data} \,. \end{cases} \tag{4.14}$$

The expensive sampling then uses the lower-dimensional latent space and yields samples r from the learned manifold. Finally, the phase space configurations are provided by the the deterministic decoder $D_\psi(r)$. Here we summarize the sampling procedure, including the energy dependence, as three sequential steps:

$$\begin{aligned} &u \sim p_\phi(u|E_\text{inc}) \\ &r \sim p_\theta(r,1|u,E_\text{inc}) \end{aligned} \tag{4.15}$$

$$x = D_\psi(r, u, E_{\text{inc}})$$

All network hyperparameters and the main training parameters are as usual given in Appendix B.

Bespoke samplers A potential drawback of CFM networks is their slower sampling than, for instance, normalizing flows with coupling layers [5] which stems from the numerical integration of the ODE in Eq.(3.58). Depending on the complexity of the target distribution, a standard ODE solver requires $\mathcal{O}(100)$ steps to achieve high-fidelity samples, each consisting of at least one forward pass of the neural network.

One method to overcome this slow inference is distillation [37,170–172], which aims to predict the sampling trajectory at only a handful of intermediate points, or even at the terminus in a single step. This requires fine-tuning the network weights using additional training time, in some cases even additional training data. Further, since the weights of the network itself are updated, consistency is not strictly guaranteed and we can end up sampling from a different distribution than was originally learned.

An alternative approach is to keep the network fixed and consider alternative structures for the ODE solver. Reference [173] provides a comparison of various training-free solvers in the context of calorimeter simulations. While training-free approaches are the least costly, they are not task-specific and therefore unlikely to be optimal. However, there exists trainable family of ODE solvers that can be optimized to a given vector field v_θ without excessive additional training [174, 175]. Such bespoke non-stationary (BNS) solvers parameterize the steps along the flow trajectory. Starting from an initial state x_0, and a time discretization $0 = t_0 < t_i < t_N = 1$, the i^{th} integration step is

$$\begin{aligned} x_{i+1} = a_i x_0 + b_i \cdot V_i \quad \text{with} \quad & a_i \in \mathbb{R},\ b_i \in \mathbb{R}^{i+1} \\ & V_i = [v_\theta(x_0, t_0), \cdots, v_\theta(x_i, t_i)] \in \mathbb{R}^{(i+1)\times d}\ , \end{aligned} \tag{4.16}$$

where we again suppress the energy dependence of v_θ. By appropriately caching the velocities, each step requires just one evaluation of the network. Including the t_i not fixed by the boundary conditions, an N-step BNS solver has a total of $N(N+5)/2+1$ learnable parameters, which is typically orders of magnitude fewer than the network v_θ. Therefore, optimizing the solver generally requires a fraction of the computation time needed to train the vector field itself. Non-stationary solvers encompass a large family of ODE solvers, including the Runge-Kutta (RK) methods. Euler's method, i.e. first order RK, corresponds to taking $a_i = 1$ and $b_{ij} = 1/N$.

Bespoke solvers can be trained by comparing the bespoke trajectory to a precisely-computed reference $x_{\text{ref}}(t)$, given an initial state x_0 sampled from the CFM latent distribution. Here we define two options. First, the global truncation error measures the deviation between the final states of the solvers

$$\mathcal{L}_{\text{GTE}} = \Big\langle [x_{\text{ref}}(1) - x_N]^2 \Big\rangle_{x_0 \sim \mathcal{N}}\ , \tag{4.17}$$

where x_N is computed by iterating Eq.(4.16) starting from x_0. The local truncation error instead measures the discrepancy at each step,

$$\mathcal{L}_{\text{LTE}} = \Big\langle \sum_{i=0}^{N-1} \left[x_{\text{ref}}(t_{i+1}) - (a_i x_0 + b_i \cdot V_{\text{ref},i})\right]^2 \Big\rangle_{x_0 \sim \mathcal{N}}\ , \tag{4.18}$$

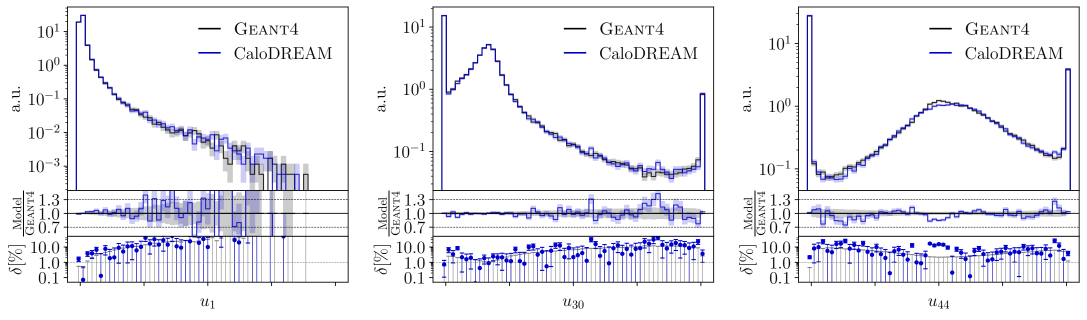


Figure 4.11: Distributions of selected u-features in DS2 from the CaloDREAM energy network (blue) compared to truth (grey). The error bars in all feature distributions in this paper show the statistics of the respective datasets.

where $V_{\text{ref},i}$ is defined as in Eq.(4.16), but with velocities evaluated on the reference trajectory.

Although we use CFM for both our shape and energy networks, we only study BNS solvers for the shape network. For training a BNS solver, we initialize it to the Euler method. At each iteration, we sample an x_0 batch from the unit Gaussian and a batch of conditions from the energy network. A precise solver is then used to generate the reference trajectory $x_{\text{ref}}(t)$ which enters the loss. Note that the shape model parameters θ are frozen during training.

4.4.2 From DS2 to DS3

Layer energies In Fig. 4.11 we compare samples generated from the energy network with the truth for a selection of normalized layer energies u_i. The transfusion network indeed generates high-quality distributions, with errors comparable to the statistical uncertainties in the test data. The distributions for $u_{i>40}$ are the most difficult to model, since the majority of showers lie in the sharp peaks at zero or one. These are zero-width peaks corresponding to showers that end at the given layer, leading to a one, or end before or skip the layer, leading to a zero.

We find that our autoregressive setup is particularly effective in faithfully mapping regions close to these peaks. As a quantitative performance measure, we train a classifier to distinguish the u's defined by our energy network from the GEANT4 truth, obtaining AUC scores around 0.51. The comparison in terms of layer energy is shown in Fig. 4.12. The factorization procedure allows us to use the same energy network for the ViT and the laViT, effectively generating statistically-identical layer energy distributions.

DS2 showers Given the learned layer energies, we use the shape networks described in Sec. 4.4.1 to generate the actual calorimeter showers over the voxels. In the first row of Fig. 4.12 we compare a set of layer-wise distributions from the networks trained in the full space and in the latent representation to the test data truth. We start with the energy deposited in layer 20, where for $E_{20} > 10$ MeV the full-dimensional vision transformer (ViT) as well as the latent-diffusion counterpart (laViT) agree with the truth at the level of a few per-cent, as expected. Towards smaller energies we see a missing feature in both networks. Also in the two other shown distributions the ViT and laViT agree with each other and deviate from GEANT4 only in regions with statistically limited training

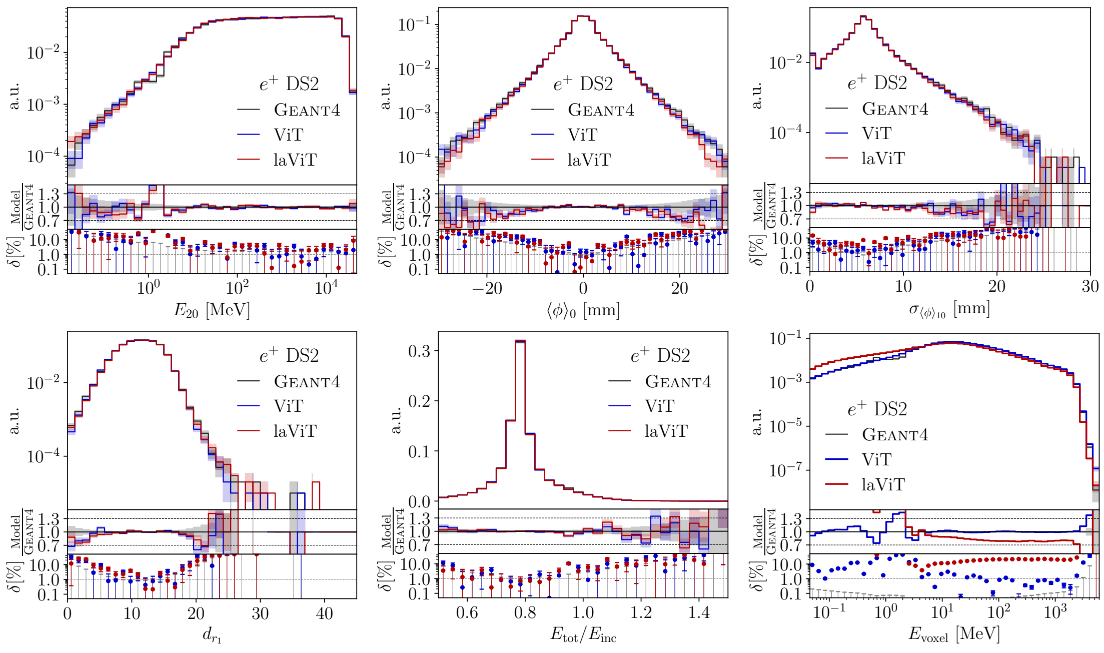


Figure 4.12: Selection of high-level features for DS2. The first row shows features for individual layers, the second row the combination of layers.

data. The second row of Fig. 4.12 shows example distributions probing the combination of layers. Here, the weighted depth of the shower highlights a small deviation for both networks from the reference for showers with maximum depth of five layers not captured by the layer-wise high-level features.

Also combining layer-wise information, we show the total energy deposited in the calorimeter E_{tot} normalized by the incident energy and the full voxel distribution across the entire calorimeter E_{voxel}. The total shower energy relative to the incident energy is reproduced very well by both networks since this information is coming from the energy network. However for the voxel energies only the full-dimensional network captures the low-energy regime, whereas the latent model overestimates this regime and in turn shifts down the prediction for larger energies because of the normalization of the curve. This is the only noteworthy shortcoming of the laViT compared to the ViT that we find.

Following up on the problem raised by the last panel in Fig. 4.12, we focus on the (latent) description with low-energy voxels. In Fig. 4.13 we again compare the two

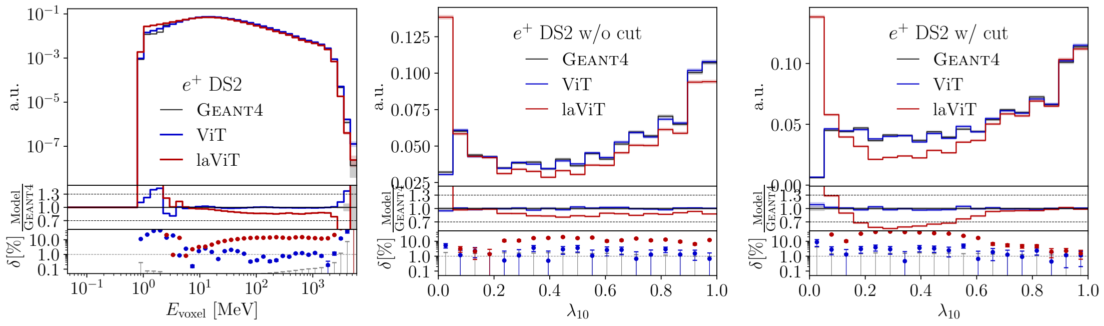


Figure 4.13: Effect of an additional threshold $E > 1$ MeV on DS2; we show the shower energy and the sparsities without and with threshold cut.

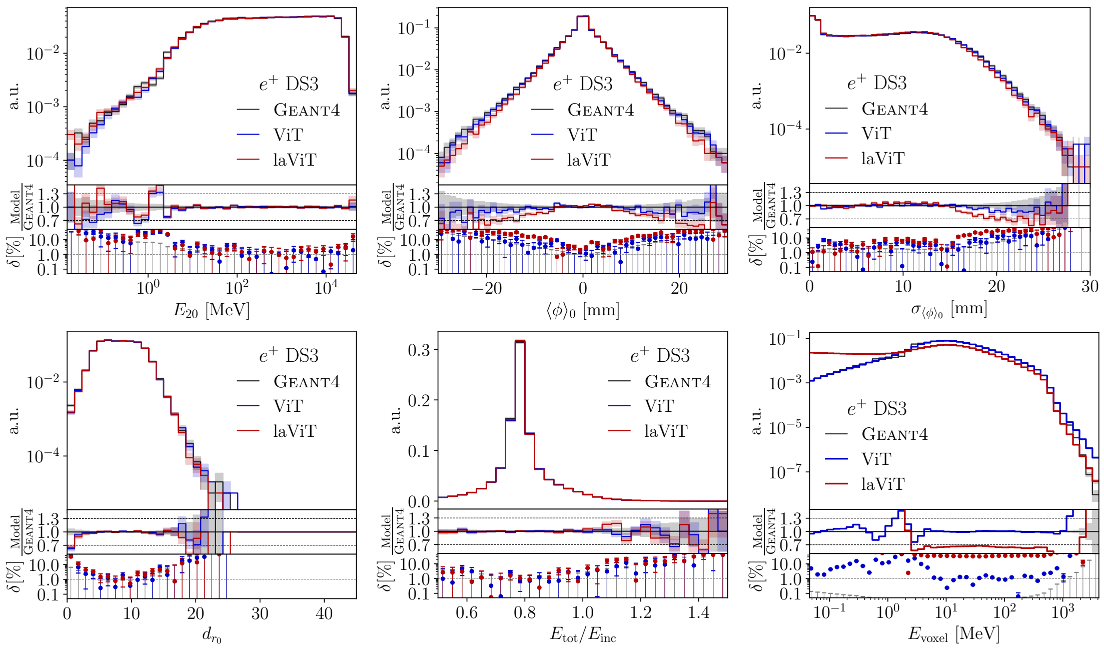


Figure 4.14: Selection of high-level features for DS3. The first row shows features for individual layers, the second row the combination of layers. All features correspond to the DS2 results shown in Fig. 4.12.

network predictions with the truth, but applying an additional threshold cut of

$$E_{\text{voxel}} > 1 \text{ MeV} . \tag{4.19}$$

After this cut, the agreement of the laViT prediction with the full ViT and the truth improves significantly. We checked that this cut has only a limited impact on the total energy deposition E_{tot}. Slight deviations are limited to the threshold region $E_{\text{voxel}} \lesssim 5$ GeV. The reason can be seen in the sparsity distributions for instance of layer 10, λ_{10}. The laViT networks generates a sizeable number of showers with energy depositions everywhere, leading to a peak at zero sparsity. This failure mode is already present in the autoencoder reconstruction as shown in Appendix A. Because of their low energy, these contributions do not affect the other high-level observables or the learned physics patterns of the showers.

DS3 showers The same analysis done for DS2 in Sec. 4.4.2 we now repeat for DS3. This means we study the same shower energies and shower shapes, but from 40500 instead of 6480 voxels. A target phase space of such large dimension is atypical for most LHC applications, and the key question is whether the precision-generative networks developed for lower-dimensional phase spaces also give the necessary precision for high-dimensional phase spaces. As a matter of fact, we know that this is not the case for standard normalizing flows or INNs [5], where the architectures have to be modified significantly to cope with higher resolution.

In Fig. 4.14 we again show a set of layer-wise features in the first row, observing extremely mild differences to the DS2 results. Only the shower shapes from the laViT suffer slightly in regions with too little training data. For the multi-layer features in the second row, we also find the same results as for DS2, including the challenge in describing voxels with $E_\text{voxel} \lesssim 3$ GeV.

Understanding and targeting this challenge, we again show the voxel energy distribution and the sparsity after the threshold cut $E_\text{voxel} > 1$ MeV in Fig. 4.15. For DS3 it turns out that after applying this cut the description of DS3 through the laViT network is excellent. The reason for this is two-fold. Given the low energy bound we can reproduce with the latent model, a cut larger than this threshold completely adjusts the sparsity up to a specific value by removing the additional energy deposition of the latent model and the noisy components of GEANT4. For both DS2 and DS3 the cut fixes the sparsity in λ_{10} up to $\lambda_{10} \gtrsim 0.7$. However, for DS3 this is done by moving the peak at zero, while for DS2 the mass is moved from the intermediate sparsity. This second difference comes from the dimensionalities of the two datasets, where the fixed reduction factor has a stronger impact on DS2 due to the larger information loss in the bottleneck.

Sampling efficiency To demonstrate the performance of bespoke samplers, we compare the quality of showers produced by various solvers in terms of classifier tests. Classifiers trained to distinguish generated and true samples are an effective diagnostic tool since they capture failure modes in high-order correlations that are hidden in simple high-level distributions. As we will see in the following section, the phase space distribution of classifier scores can be used to search for and identify such failure modes. In this section, we only use the AUC as a simple, one-dimensional quality measure. The high-level classifier uses the layer-wise features but since we want sensitivity also to voxel-level correlations, we train a classifier on the low-level phase space as defined by the original voxels.

For our comparison, we include three standard fixed-steps solvers: the Euler, Midpoint, and Runge-Kutta 4 methods. We also consider bespoke non-stationary solvers using either the global, Eq.(4.17), or local, Eq.(4.18) truncation error as described in Sec. 4.4.1. Using each solver, we generate 100k showers from the DS2 ViT shape network. We train classifiers to distinguish these samples from the GEANT4 reference set using the standard CaloChallenge pipeline. In Fig. 4.16, we plot the high-level (left) and low-level (right) AUC scores against the number of function evaluations n_eval for each solver. Note that the Midpoint and RK4 methods respectively use 2 and 4 function evaluations per integration step. See App. 4.4.2 for function evaluation timings of each network.

In both panels of the figure, we see that the Euler solver has a notably poor efficiency in terms of function evaluations. This indicates that the velocity field learned by the

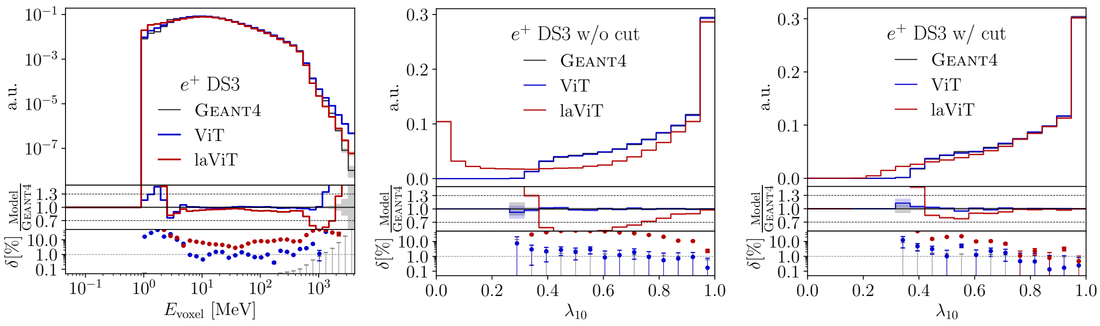


Figure 4.15: Effect of an additional threshold $E > 1$ MeV on DS3; we show the shower energy and the sparsities without and with threshold cut. All features correspond to the DS2 results shown in Fig. 4.13.

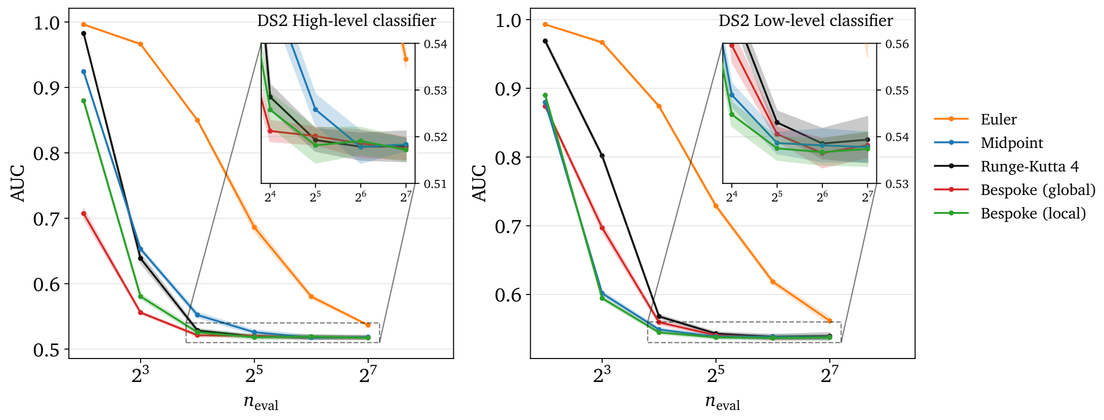
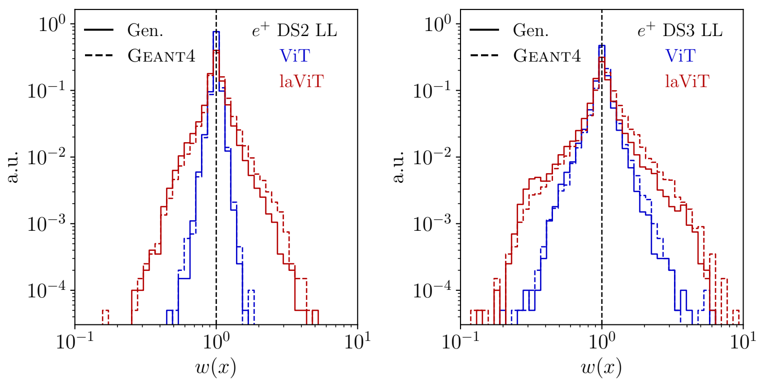


Figure 4.16: High-level (left) and low-level (right) classifier AUC scores on DS2 as a function of the number of function evaluations n_{eval} for various ODE solvers. Samples of 100k reference and generated showers are used to train the classifier. Errors bands are taken as the standard deviation over 10 runs.

shape network has non-trivial curvature. Considering the remaining solvers, the sample quality essentially saturates by $n_{\text{eval}} = 64$ and all non-Euler methods appear to have statistically-equal performance at this point. The bespoke samplers demonstrate the best retention in quality when looking toward smaller n_{eval}. In particular, the local BNS solver keeps an AUC below 0.6 for both classifiers even at 8 function evaluations. The global BNS solver achieves a large margin of improvement at $n_{\text{eval}} = 4$ for the high-level classifier. The local bespoke solver also shows an advantage in the high-quality regime. Specifically, its AUC is already saturated for both solvers at 32 function evaluations. As such, in a resource-limited scenario the efficiency gains offered by bespoke solvers can be translated into improved sample quality.

It is interesting to note that the performance of a given solver can be significantly different between high- and low-level classifiers. This is evident in the reversed rankings of, for example, the two bespoke solvers in each panel. The global BNS solver favors performance on the high-level classifier, while the local BNS solver is best on the low-level classifier. A similar exchange can be seen among the Midpoint and RK4 solvers, with the former being close to optimal at low level.

Performance In Fig. 4.17 we show the classifier weights from the low-level classifier for DS2 and for DS3. We also include a table with the AUC scores of the high-level classifier trained on layer-wise features and the low-level classifier, where the ViT shows state-of-the-art results on DS2 and the high-level DS3. The peaks of the weight distributions are nicely centered around $w = 1$, symmetric towards small and large (logarithmic) classifiers, and show no significant difference between generated and training data. The weights for the networks encoding the full phase space and the latent diffusion are different, with a typical broadening of the distribution by a factor two around the peak and larger and less smooth tails. We still observe that the classifier misses the low-energetic noise affecting the sparsity and the voxel energy distributions. Despite the simple nature of the neural network, a sequence of fully connected layers, the main result from this performance test is that the classifier identifies additional failure modes related to the step from DS2 to DS3 and to the reduced latent space. We expect these failure modes correspond to cross-layer features, since we observe a correlation between the classifier weights and the



	AUC (LL/HL)	
	DS2	DS3
ViT	0.54/0.52	0.63/0.53
laViT	0.58/0.53	0.62/0.59

Figure 4.17: Learned low-level classifier weight distributions for DS2 (left) and DS3 (right). We compare the full-dimensional ViT and the latent laViT results and, for each of them, show weights for the generated sample and for a GEANT4 test sample. The table shows the AUC values for the trained classifier in each case.

Network	Time (ms)	
	DS2	DS3
Energy	0.37±0.01	0.37±0.01
Shape (ViT)	17±1	84±8
Shape (LaViT)	31±1	63±6

Table 4.3: Timings for one network forward pass using batch size 100 on an NVIDIA H100.

shower depth introduced in Sec. 4.4.2, and the high-level AUC is similar across the two datasets. Details of the neural network classifier are listed in Appendix B.

Timing In Sec. 4.4.2, we study the sampling cost of networks in terms of the number of function evaluations n_eval. Here we provide timing measurements for a single forward pass of each of our CFM networks, using a batch size 100. We ran tests using a single NVIDIA H100 GPU and summarize the results in Tab. 4.3. The times for the energy network are identical across the two datasets since there is no change in the network architecture. Also note that since the energy model is autoregressive, sampling with an N-step solver uses $N \times L$ function evaluations, where L is the number of calorimeter layers.

4.5 Understanding generative networks

Finally, we showcase the power of classifiers to understand failure modes in generative networks starting from weight distributions. As an example of how to use weights over phase space, we turn to the classic calorimeter simulation. The dataset used for this study was generated in a previous iteration of fast surrogate models [12, 14, 21, 23]. We study weight distributions for positron, photon, and pion showers in a simplified calorimeter. The classifier defined in Appendix B is trained on voxels, energy, and layer energies. We

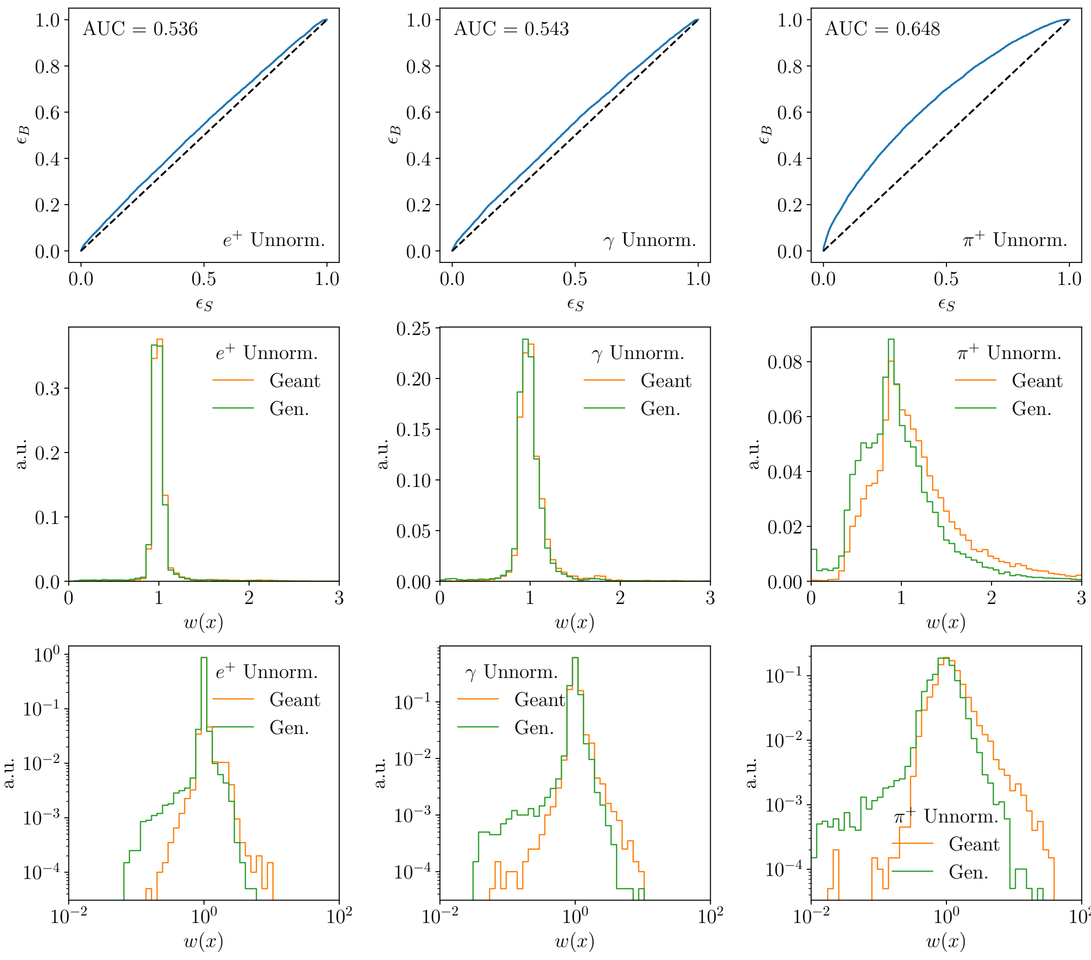


Figure 4.18: Left to right: calorimeter showers for e^+, γ, and π^+. Top to bottom: ROC curve, weight distribution on a linear scale, and weight distribution on a logarithmic scale. The weights are evaluated separately on the Geant dataset used for generator training and the generated dataset.

focus on the classifier with unnormalized preprocessing in this work because it appears to be better calibrated and shows less propensity for overfitting. For more discussion, see Sec. 4.5.3.

4.5.1 Tails of weights

In Fig. 4.18 we show the receiver operating characteristic (ROC) curves and weight distributions for e^+, γ, and π^+ showers generated from the CaloINN and compared to the reference dataset. The top row confirms that positron and photon showers are easier to generate than pion showers. The question is which potential failures are related to this performance difference.

In the second row we show the weight distributions. First, we observe that they are not symmetric, because the reweighting now compensates features. The limit $w(x) = 0$, most visible for the pion shower, marks phase space points where the generator has learned a finite density $p_\theta(x)$, where the correct density is $p_{\text{data}}(x) = 0$.

In the third row of Fig. 4.18 we show the same curves on a logarithmic scale to see the tails. As expected, they are different when evaluated on GEANT and generated showers. Already for positrons, the generated data includes many more showers with $w(x) \ll 1$ than the training data. These are showers for which the generator overpopulates phase space, so they appear preferably in the generated dataset. This tail connects to showers with weight zero.

In contrast, showers with $w(x) \gg 1$ appear more frequently in training dataset. These under-populated regions of phase space correspond, for instance, to features or tails which the network does not learn. This serious failure mode can be identified by evaluating showers with anomalous weights on the training data.

4.5.2 Phase space clustering

The simpler structure of photon showers allows for a detailed study of the clustered observables. By cutting on the weight values and looking at the distribution of the remaining photon showers, we identify three characteristic failure modes highlighted with different colors Fig. 4.19.

1. In orange, we isolate the large-weights tail with $w > 1.6$ and no energy deposited in layer 2 ($E_2 < 0.1\,\text{MeV}$), as shown in Figs. 4.19(c) and 4.19(f). As shown in Figs. 4.19(g) and 4.19(h), these showers have higher sparsity in layers 0 and 1 than the typical shower. Additionally they have lower energy, shown by the E_1 histogram in Fig. 4.19(e), since on average most of the energy is deposited in layer 1. Overall, these showers consist of just a few activated, low-energy voxels in layers 0 and 1, and exactly none in layer 2. This sub-population of showers exists in the GEANT data, but it is not sufficiently generated by the network.
2. In blue, we isolate the small-weights tail with $w < 0.6$. Fig. 4.19(c) shows that this failure mode is characterized by a single voxel carrying all the energy in layer 2, and Fig. 4.19(e) shows that this energy is lower than the average energy deposition. Blue and orange agree in every feature that we looked at in layers 0 and 1; they only differ in layer 2. Since these are showers overproduced by the generator, we interpret this as the compensation of the generator for the underproduction of the orange showers; the compensation is only needed in layer 2. We think the reason for both the orange and blue failure modes is due to the low energy and the large number of zero voxels in these showers: this causes them to be especially sensitive to the noise we add during training, since a single voxel is being activated and it either falls just under or just over the minimum energy threshold. The vicinity of these showers to the noise threshold makes it harder for the generator to perfectly model this region of phase space.
3. Finally, in green we isolate again the large-weights tail with $w > 1.6$ that *does* deposit energy in layer 2 ($E_2 > 0.1\,\text{MeV}$). These showers are also underproduced by the generator but they are distinct from the previous two classes. According to Fig. 4.19(d)-(f), these have very low energy in layer 0 (even lower energy than the orange showers), and higher-than-typical energy in layer 2. In layer 1 their energy is closer to the typical shower. We also see in the sparsity that these photons deposit very little energy and activity in layer 0, while in layers 1 and 2 they are fairly typical. These are showers which develop late in the calorimeter, leaving little or no energy in layer 0. Interestingly, physics tells us that these late-developing showers

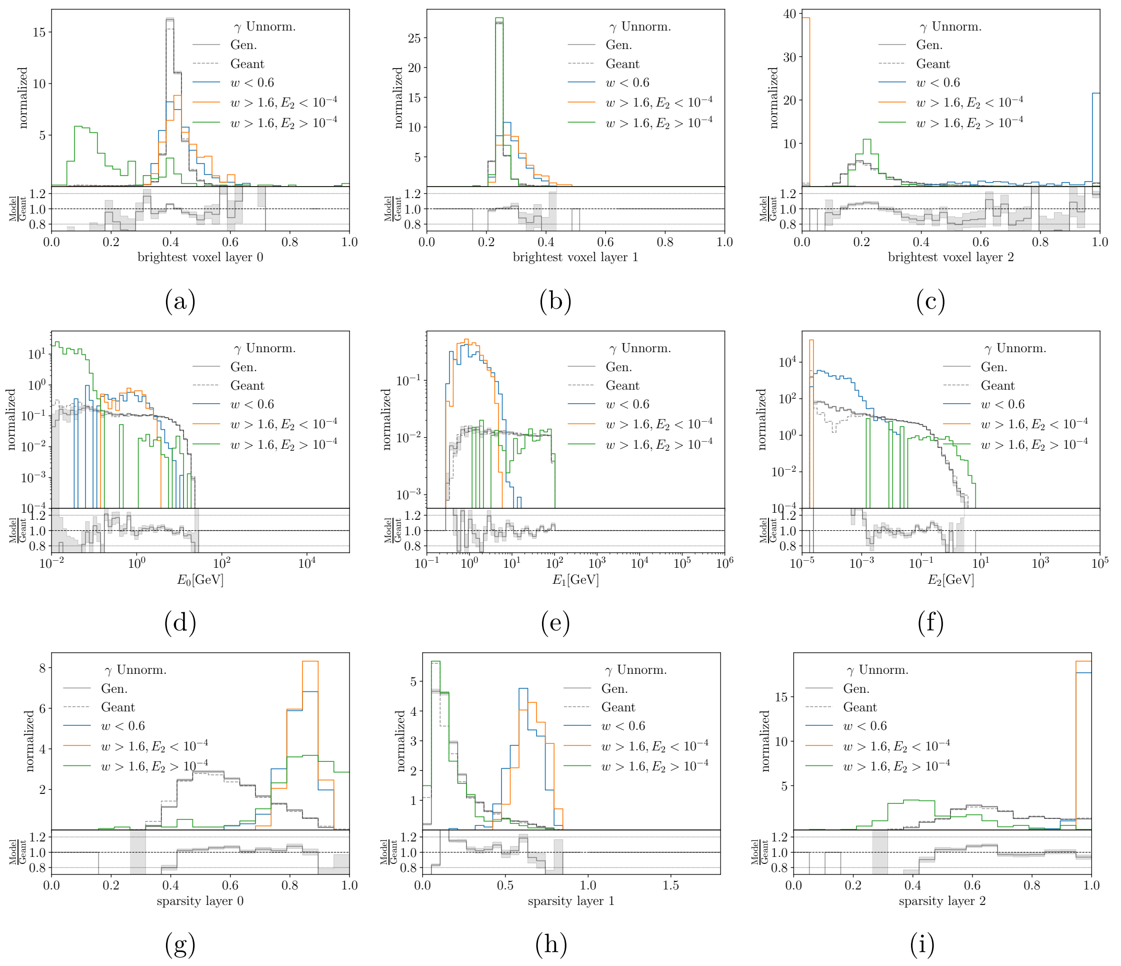


Figure 4.19: Relevant distributions for γ showers in the small-weights (blue) and large-weights regions (orange and green). We show the energy depositions, the fraction of the energy deposited in the leading voxel, and the sparsity in the three layers of the calorimeter.

are possible for photons but not likely for positrons. At high energies, the latter interact continuously with the material through Bremsstrahlung, while the former need to convert to e^+e^- first [176]. This leads to showers fully absorbed deeper in the calorimeter, therefore with more energy deposited in layer 2. We see this difference in the physics clearly reflected comparing with the green showers for the positron case. The positrons have energy deposited in layer 0, unlike the photons.

The situation becomes much more complicated when looking at pions, where the more complex physics through the nuclear interaction and the poorer generative model make it harder to identify failure modes with kinematic or physics features. In line with the sobering AUC value given in Fig. 4.18, we see in Fig. 4.20 that the generator requires correction weights essentially all over phase space. The first distinctive failure mode is corrected by small weights in the energy distributions, for instance in layer 0, which suppress the generated showers to reproduce the sharp lower edge of the energy deposition. In addition, the network produces too many showers with exactly zero energy deposition in layers 1 and 2. They are included in an overflow bin in the energy histograms, but appear as a failure mode in the energy fraction of the brightest voxel, for example in

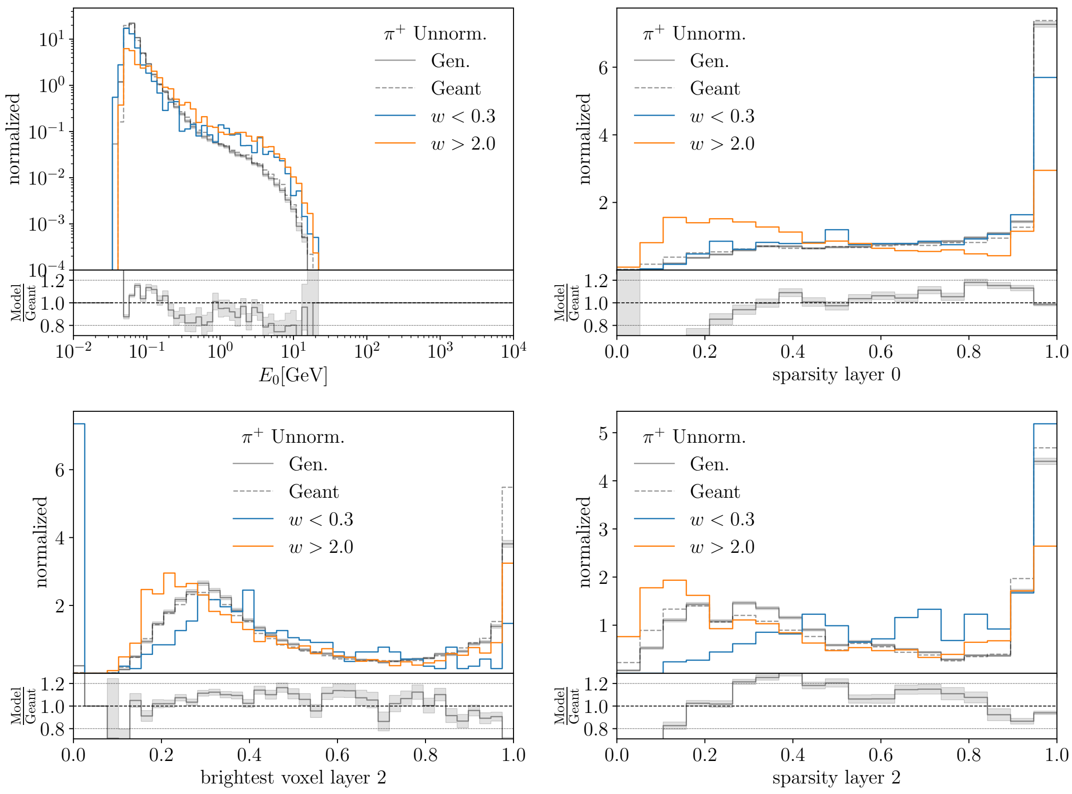


Figure 4.20: Relevant distributions for π^+ showers in the small-weights (blue) and large-weights regions (orange). We show the energy deposition and the sparsity in layer 0, and the brightest voxel energy and the sparsity in layer 2.

layer 2. Finally, we see showers with large weights cluster at low sparsities. Here the generator has a systematic bias towards simpler showers with fewer voxels. Given these observations, the leading improvement to the generative model concerns the low-energetic voxels. As discussed before, this can be linked to the addition of noise during training and provides us a research direction to improve the generator, e.g. sampling from a different noise distribution or the development of a noise-less training scheme. We provide the full set of studied histograms in Appendix A.

4.5.3 Classifier calibration

To gauge whether the classifiers used in our study have been well-trained (not overfitted, reasonably close to optimal), one important check is to inspect their calibration curves. The idea of the calibration curve is that a properly learned and optimal classifier $C(x)$ should return the probability that x is class 1, and $1 - C(x)$ the probability that x is class 0. Therefore, if we took all events x in the training data (assumed to be balanced) for which $C(x) = C$, a fraction C of them should be class 1. The differential way to write this is

$$\frac{\frac{\mathrm{d}N_1}{dC}}{\frac{\mathrm{d}N_1}{dC} + \frac{\mathrm{d}N_0}{\mathrm{d}C}} = C \; , \tag{4.20}$$

where N_i is the number of events in class i. As in the other sections, we will look at calibration curves in terms of the weights w. Using Eq.(4.6), we can turn Eq.(4.20) into a statement about the weights,

$$\frac{\mathrm{d}N_1}{\mathrm{d}w} = \frac{\mathrm{d}N_0}{\mathrm{d}w}\, w\ . \tag{4.21}$$

Equation (4.21) implies an equivalent way of plotting a calibration curve in weight space: divide the combined weight distribution in bins and calculate the ratio $N_{\mathrm{truth}}/N_{\mathrm{gen}}$ for each bin. According to Eq.(4.21), for a well-calibrated classifier these should agree. We show calibration curves, calculated following this method, for our classifiers in Fig. 4.21. We see that the classifiers are for the most part very well-calibrated. One possible exception is for e^+ at lower weights, but one should keep in mind this is one of the better generative models considered in this work (AUC=0.536), so nearly all the events are in the well-calibrated part of the calibration curve (with $w \approx 1$). Also, as we see in the discussion in Sec. 4.5.2 and in Fig. 4.19, even if the tails of the classifier are mis-calibrated, it can still extract poorly modeled regions of phase space and assign, if anything, too extreme weights to them.

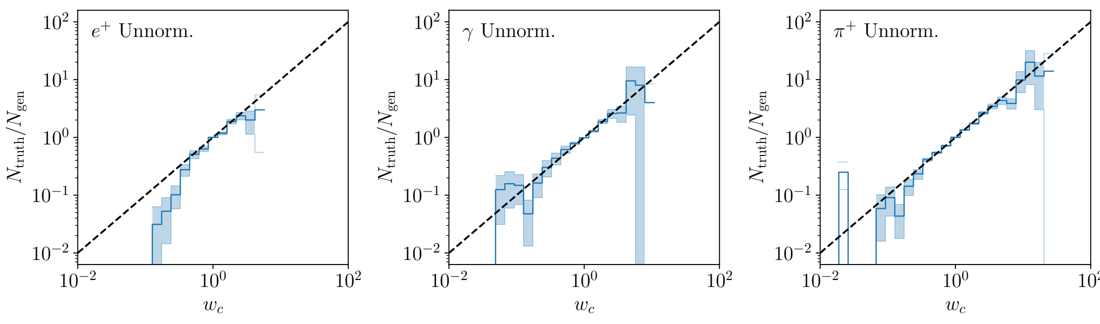


Figure 4.21: Calibration plots in weight space for the different discriminator models. From left to right, normalized showers for e^+, γ, and π^+.

CHAPTER **5**

Model agnostic searches

The content of this chapter was finalized in collaboration with Barry M. Dillon, Friedrich Feiden, Michael Krämer, Tanmoy Modak, Tilman Plehn, Jan Rüschkamp, and Peter Sorrenson.

The big goal of the LHC is to discover physics beyond the Standard Model and to identify new properties of the fundamental constituents of matter. Until now, we pursue BSM searches based on pre-defined theory hypotheses. However, the lack of a clear direction for future searches motivates a more model agnostic approach, where we look for anomalies in the data. This chapter starts with the datasets and the representations we used for our studies. Following, we study anomaly detection using autoencoders. We introduce an energy-based network, the normalized autoencoder (NAE), that solves the complexity bias in autoencoders. We describe the NAE setup, with its efficient way of sampling the background data manifold in phase space and latent space. In Sec. 5.2.1 we apply the NAE to the top tagging dataset [177–179] and show that, for the first time, the NAE identifies anomalous top jets and anomalous QCD jets symmetrically and with high efficiency. Next, we target two challenging dark jet signals [58] and confirm the excellent performance of the NAE and its relative independence of the jet image preprocessing in Sec. 5.2.3.

In the second part of the chapter, we show the importance of learning powerful representations of the data for anomaly detection. We develop a new approach to density-based anomaly detection using self-supervision, which defines the representation of the data in a model-agnostic way using the power of highly expressive networks such as transformers or graph networks to boost anomaly detection. We introduce AnomalyCLR and DarkCLR, both methods based on the idea of 'anomaly-augmentations'. These anomaly-augmentations are modifications of the original event to which the underlying physics is not invariant. In fact these augmentations are chosen to mimic very general features that anomalous events might have, such as high multiplicity, large missing transverse energy (MET), or large p_T. Despite choosing explicitly the augmentations, the approach does not target any specific new physics model, and we will see from the results that the approach is model agnostic.

In Sec. 5.1 we will discuss the dataset and the different backgrounds and signals. In Sec. 5.3 we introduce the AnomalyCLR idea, first discussing contrastive learning and then how this can be modified for use in anomaly detection. The specifics of the application to event-level collider data and jet substructure is given in Sec. 5.3.2. The discussion on how we estimate anomaly scores is given in Sec. 5.3.4, where the architecture and optimisation of the autoencoders we use is discussed. The results are presented in Sec. 5.3.5 and Sec. 5.3.6, along with an analysis of how different anomaly-augmentations and different representation dimensions affect the results.

5.1 Events and jet substructure

Robustness is a fundamental requirement for an anomaly detection method. We test the robustness towards different signals and data formats by testing the networks on several datasets which are described in this section.

5.1.1 Benchmark physics models

Jet constituents Jets are a prevalent signature of several new physics models, the first test will be top vs QCD, discussed in Sec. 5.2.1, where we use the top-tagging dataset [177–179], also used for the AE in Ref. [139] and the Dirichlet VAE in Ref. [180]. We start with anti-k_T jets [181] with $R = 0.8$, defined by FastJet3.1.3 [182] as substructure containers. The top and QCD jets are are required to have

$$p_T = 550 \text{ ... } 650 \text{ GeV} \qquad \text{and} \qquad |\eta| < 2 \, . \tag{5.1}$$

This task is considered a benchmark test for any tagging methods. In the boosted regime top jets produce a typical three-prong jet substructure due to the decay products being clustered in a single fat jet.

A second class of challenging signatures are Hidden Valley models, which can lead to tantalizing semi-visible jets at the LHC. In Sec. 5.3 and Sec. 5.3.6 we are interested in Hidden Valley models that consist of a strongly coupled dark sector with dark quarks coupled to the Standard Model (SM) through a vector mediator. As a result, jets can be produced by the dark quarks from the decay of the vector mediator. The shower in this case would involve radiation into the dark sector, resulting in jets that are called semi-visible or dark jets, depending on the phenomenology of the signal.

For our purposes, we consider a benchmark signal scenario with an underlying dark sector as introduced in [58, 183, 184]:

$$pp \to Z' \to q_d \bar{q}_d, \quad \text{with} \;\; m_{Z'} = 2\,\text{TeV} \;\; \text{and} \;\; q_d = 500\,\text{MeV}, \tag{5.2}$$

where Z' is the mediator between the dark sector and the SM quarks, charged under a $U(1)'$ gauge group, and q_d is a dark quark charged under a dark $SU(3)_d$. The dark sector hadronizes to dark pions ($\pi_d = 4$ GeV) and dark rho mesons ($\rho_d = 5$ GeV). The neutral dark rho mesons mix with the Z' and can thus decay into SM quarks. The other dark mesons are stable and escape detection. In our benchmark scenario the fraction of invisible particles in a shower is given by $r_{\text{inv}} = 0.75$ [183, 184]. This dark sector model then leads to semi-visible jets and can be simulated with the Pythia Hidden Valley module [185, 186]. We will refer to this benchmark scenario as the "Aachen" dataset in the remainder of the paper.

The dataset is generated using Madgraph5 [187] for the hard process. The generated events are then interfaced with Pythia 8.2 [188] for showering and hadronization and finally fed to Delphes 3 for fast detector simulation [189]. The jets are reconstructed using the anti-k_T algorithm [190] with radius parameter $R = 0.8$ in FastJet [182].

The most important phenomenological parameters for Hidden Valley models are the invisible fraction of the constituents, r_{inv}, and the mass of the dark mesons, $m_{\pi/\rho}$. To test the model dependence of our approach, we generate several data sets with the following parameter choices: starting from our benchmark signal, we first vary only the mass of the dark mesons and the confinement scale Λ as $m_{\pi_d} = m_{\rho_d} = \Lambda = 10\,\text{GeV}$, $20\,\text{GeV}$. In addition, for our default choice of dark meson masses, we change the invisible fraction r_{inv} by allowing all dark mesons to decay back to SM quarks with a given probability. To explore the region where the number of visible jet constituents is closer to the QCD background, we reduce the invisible fraction to $r_{\text{inv}} = 0.5$, 0.2. The light QCD background is generated from leading order di-jet events. A second model that leads to a modified QCD jet [139] is also used as a reference and called "Heidelberg" signature in the following sections.

The selection of the jets at detector level is done by calculating the ΔR between the reconstructed fat jets and the dark quarks at parton level and ensuring that $\Delta R < 0.8$. On the selected fat jets we apply a kinematic selection in p_T and η, namely

$$p_T^j = 150...300 \text{ GeV} \quad \text{and} \quad |\eta^{\text{j}}| < 2\,. \tag{5.3}$$

Reconstructed events To test the performance of applying representation learning to raw data in a simpler scenario for an anomaly detection task we use the CMS anomaly detection challenge dataset [191], which contains simulated proton-proton collisions with a 13 TeV centre-of-mass energy. We will refer to this network as AnomalyCLR and it is discussed in Sec. 5.3. Differently from the jet constituents, this dataset contains reconstructed objects in an event resulting in a much lower input dimensionality. The events are selected to have at least one e or μ with transverse momenta $p_T > 23$. The pseudo-rapidity ($|\eta|$) is required to be < 3 and < 2.1 respectively for e and μ. Further, the events are allowed to have up to 10 jets with $p_T > 15$ GeV and $|\eta| < 4$, up to 4 muons $p_T > 3$ GeV and $|\eta| < 2.1$, up to 4 electrons $p_T > 3$ GeV and $|\eta| < 3$ and missing transverse energy (MET). The dataset is generated with Pythia 8.240 generator [188] with a fast detector simulation using Delphes 3.3.2 [189] with the Phase-II CMS detector card. The jets are reconstructed using anti-k_t algorithm [190]. In the provided dataset each event is formatted such that the first entry is assigned for MET, next eight are assigned for electrons and muons respectively and, the final 10 entries are for jets. For each particle object the data set contains information of p_T, η, ϕ and particle id such that the shape of an event in the data frame is [N,19,4] where N is the total number of events. Note that if an event has less than the maximum allowed of a type of object, the remaining entries in that case are zero padded. The background dataset consists of a number of Standard Model processes and to determine the performance of the anomaly detection algorithm four light BSM scenarios are considered.

For the SM background a collection of events are generated from production channels with at least a single lepton in the final state. The fraction of events to be included in the SM for each process is fixed by its trigger efficiency and the LO cross section. Thus, four leading processes are considered: W and Z inclusive productions, QCD multijet contributions, and $t\bar{t}$ production. The proportions between the four processes are given in [192] as:

$$\begin{aligned}
&pp \to W^{\pm} + \text{jets} \to \ell^{\pm}\nu_\ell + \text{jets} && (59.2\%)\\
&pp \to Z + \text{jets} \to \ell^{+}\ell^{-} + \text{jets} && (6.7\%)\\
&pp \to t\bar{t} + \text{jets} && (0.3\%)\\
&pp \to \text{jets} && (33.8\%)\,. \qquad (5.4)
\end{aligned}$$

with $\ell = e, \mu, \tau$. The QCD multijet production is by far the largest production process at the LHC. Although leptons in QCD multijet backgrounds are rarely present and mainly originate from decays of unstable hadrons, the sheer volume of QCD multijet production makes it one of the largest processes in the data stream for the challenge.

The signal datasets provided by the challenge consist of events simulated from the following signal models:

- **Leptoquark (LQ)**: A 80 GeV LQ decaying in to a b and τ.
- **Neutral scalar boson** A: A 50 GeV neutral scalar boson A. The production mechanism
  $pp \to A + X \to Z^* Z^* + X$ (with X is inclusive activity) followed by both Z^* decaying into charged leptons.
- **Scalar boson** h^0: A scalar boson 60 GeV h^0 with $pp \to h^0 + X \to \tau^+\tau^- + X$ production.
- **A charged scalar** $h^{\pm}$: Charged scalar with 60 GeV mass and $pp \to h^{\pm} + X \to \tau\nu + X$ production.

The most distinguishing high-level features of these signals when compared with the background processes are the electron, muon, and jet multiplicities and the p_T and MET distributions.

5.1.2 Representations

Jet images As our first application we choose jet images. Before defining the jet image, the jets are pre-processed by centering each jet around the k_T-weighted centroid of all constituents. Then, the jets are rotated such that their principal axis points to 12 o'clock, and flipped so that the highest p_T region is in the lower-left quadrant. Then, the constituents are pixelized in 40×40 images with pixel size $[\Delta\eta, \Delta\phi] = [0.029, 0.035]$. The intensity of the pixels is defined by the sum of p_Ts within that cell, and finally, the whole image is rescaled by the total p_T of the event. To reduce the sparsity we apply a Gaussian filter to each image. The effect of two Gaussian filters is illustrated in Fig. 5.1. Our top tagging dataset consists of 140k jets for each class, of which we use 100k jets for training, and the remaining 40k for testing. The jet images are pre-processed with a Gaussian filter with $\sigma_\text{G} = 1$.

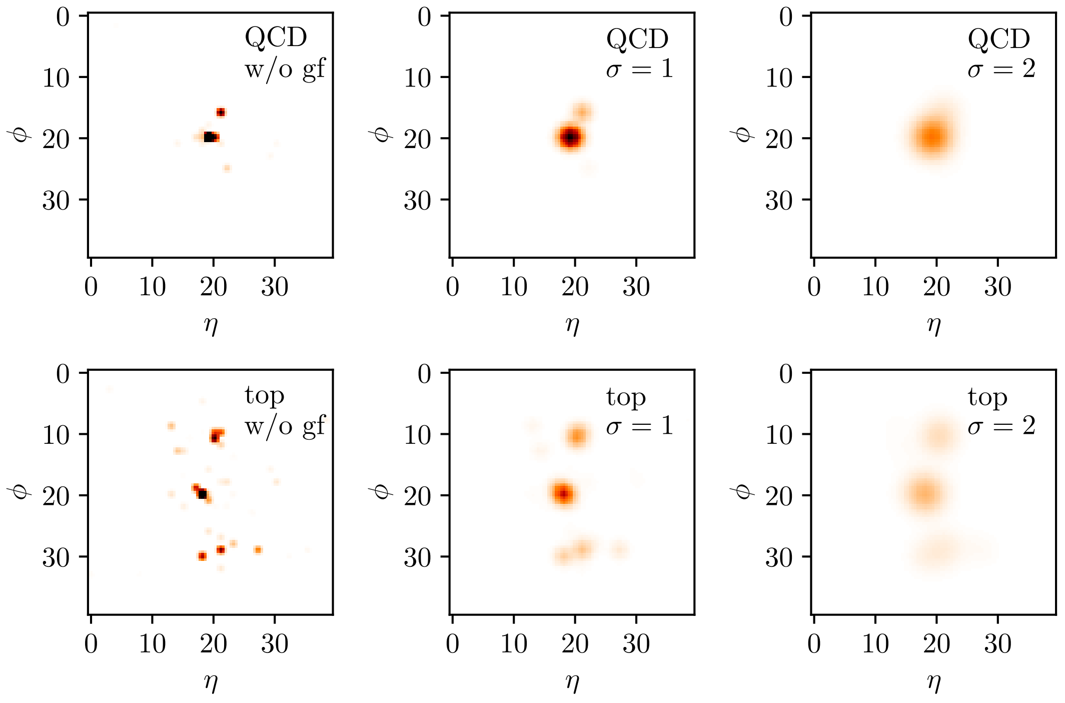


Figure 5.1: Example QCD and top images without Gaussian filter, $\sigma_\mathrm{G} = 1$, and $\sigma_\mathrm{G} = 2$.

Our second reference dataset, presented in Sec. 5.2.3 are the two dark-matter-inspired signal samples [58]. Compared to the QCD background, the Aachen dataset is mostly more sparse, whereas the Heidelberg dataset includes an additional decay structure. As for the top jets, a Gaussian filter improves the network training and we use a filter with a $\sigma_\mathrm{G} = 1$, but some additional precautions are needed for dark jets. We know that for an efficient identification of both of the dark jets we need to reweight the jet images. Unlike in Ref. [58] we now apply the same pixel-wise remapping for both dark jet signals, namely

$$p_T \quad \to p_T^n \qquad \text{with} \qquad n = 0.01, 0.1, 0.2, 0.3, 0.5 \; . \tag{5.5}$$

The goal is to reduce the dependence of an autoencoder performance on this remapping for different signals.

Point-clouds Although images give an interpretable visualization of a jet, the final input is a rather sparse representation where only a handful of pixels are activated. A more natural approach is to consider the jet as a point-cloud. In this case one data instance is a set of objects of the form $\{(p_T, \eta, \phi)\}_{i=1}^N$, where N is the number of constituents of the jet. We ignore the mass of the particles because it is set to zero in our datasets. In this representation we use the lowest level information available in a detector after reconstruction to maximize the sensitivity of the anomaly search. We set the maximum length of the jet zero-padding the vector if smaller jets are encountered. Although we have zeros in the padded input features, we develop an architecture that is totally insensitive to the contribution coming from these constituents. We adopt this representation for the studies in Sec. 5.3.

5.2 Normalized autoencoder

The normalized autoencoder [193] we will use for this study is an energy-based modification of a standard AE, as applied in Ref. [139]. We already discussed in Sec. 3.2 the featurs of autoencoders but also the problems from the vanilla implementation. However, we can upgrade the AE to a probabilistic NAE by using the MSE as the energy function in Eq. (3.30)

$$E_\theta(x) = \text{MSE} \equiv \frac{1}{N} \sum_{\text{pixels}} |x - f_\theta(x)|^2 \ . \tag{5.6}$$

This way we can train a probabilistic AE using Eq. (3.34) and Eq. (3.35).

The NAE training includes two steps. First, we pre-train the baseline AE with the standard MSE loss, similar to Ref. [139]. After the AE pre-training we switch to the NAE loss given in Eq. (3.30). All NAE parameters are given in Tab. 5.1. In the spirit of a proper anomaly search we use the same network and hyper-parameters throughout this paper. They reflect a trade-off between sampling quality, training stability, and tagging performance.

The pre-training phase builds an approximate density estimator exclusively based on the training data by minimizing the reconstruction error. Then, the NAE loss explores the regions with low energy and guarantees the behavior of the model especially in the region close to but not in the training data distribution. Here, the mismatch between the data and the model distribution is corrected by the inter-play between the two components of the loss function. We cannot give such a guarantee for a standard autoencoder, which only sees the training distribution and could assign arbitrary reconstruction scores to data outside this distribution. In fact this is the source of the problems with standard autoencoders outlined in the introduction, which are solved using the NAE.

As mentioned above, training EBMs is a practical challenge. Several algorithms have been proposed to train these networks. Two well-known methods based on MCMC samples are contrastive divergence (CD) and persistent CD. CD and PCD differ in how they define the initial sample. CD uses a sample taken from the data distribution $p_{\text{data}}(x)$ while PCD samples from a replay buffer made up of the final state of Markov chains from previous steps of the optimization. However, these methods are susceptible to creating spurious high density modes and struggle with full space coverage [193].

We follow a different approach, using the fact that we can train a regular autoencoder before starting the NAE training. If we accept that different initializations of the MCMC defined in Eq. (3.30) lead to different results, we can tune λ_x and σ_x in such a way that we can use a sizeable number of short, non-overlapping Markov chains [194–196]. Specifically, the proposed algorithm for an efficient training of NAEs is on-manifold initialization (OMI) [193]. This approach is motivated by the observation that sampling the full data space is inefficient due to its high dimensionality, but the training data lies close to a low-dimensional manifold embedded in the data space x. All we need to do is to sample close to this manifold. Since we are using an autoencoder this manifold is defined implicitly as the image of the decoder network, meaning that any point in the latent space z passed through the decoder will lie on the manifold. This means we can first focus on the manifold by taking samples from a suitably defined distribution in the low-dimensional latent space, and then map these samples into data space via the decoder. After that, we perform a series of MCMC steps in the full ambient data space to allow the Markov chains to minimize the loss around the manifold.

To sample from the model we first need to define a suitable latent probability density, which we do as

$$q_\theta(z) = \frac{e^{-H_\theta(z)}}{\Psi_\theta} \qquad \text{with} \qquad H_\theta(z) = E_\theta(f_D(z)) \; , \tag{5.7}$$

where $H_\theta(z)$ is the latent energy, and f_D is the decoder network, all in complete analogy to Eq. (3.29). Having defined these quantities, the latent space chain is run as

$$z_{t+1} = z_t + \lambda_z \nabla_z \log q_\theta(z) + \sigma_z \epsilon_t \qquad \text{with} \qquad \epsilon_t \sim \mathcal{N}_{0,1} \; . \tag{5.8}$$

Once we reach a high-density point on the decoder manifold, the final sample is obtained by running a second input chain according to Eq. (3.35).

During the OMI it is crucial that we cover the entire latent space, thus a compact structure is preferable. To achieve that, we normalize the latent vectors so that they lie on the surface of a hypersphere $\mathbb{S}^{D_z-1}$, allowing for a uniform sampling of the initial batch in the latent space. The step size and the noise of both chains are tuned to give $T < 1$. Even if a lower temperature introduces a bias towards modes with lower reconstruction error, this helps stabilize the training and obtain finer samples from the MCMC. Long LMC chains are affected by instability by two reasons: sudden changes in the gradients between steps, and diverging energy for both positive and negative samples due to the loss function being independent of constant shifts. Even if these issues are still not well understood in the ML community, different possible solutions have been proposed [194] and applied in this work: (i) clipping gradients in each step; (ii) spectral normalization; (iii) L2 weight normalization; and (iv) L2 normalization on positive and negative samples. To decrease the size of both chains, a replay buffer has been utilized which saves the final points of each latent chain. In the next iteration the initial sample is either drawn from the buffer or drawn uniformly from the hypersphere, with the probability of being drawn from the buffer being 0.95. Finally, an acceptance Metropolis step and a noise annealing step can be applied.

The encoder has 5 convolutional layers. Each layer has 8 filters, except for the last layer with one filter. The output is then flattened, and two dense layers downsize the network to the latent space size. The decoder mimics the encoder with 2 dense layers followed by 4 convolutional layers. All intermediate activation functions are PReLU. The output activation for the encoder and the decoder are linear and sigmoid, respectively. For the latent space dimension we use $D_z = 3$, which is not optimized for performance, but allows us to visualize the latent space easily. We run the pre-training for 300 epochs, using Adam [134] with default parameters. Additional information on the network architecture can be found in Appendix B.

5.2.1 QCD vs top jets

A simple, established anomaly detection task based on jet images is to extract top jets out of a QCD jet sample, with network training on background only [139]. We summarize the results with a focus on the performance for symmetric tagging of QCD vs top images. In standard autoencoder networks a known problem is that they tend to assign larger reconstruction losses to samples with higher complexity rather than those which are not well-represented in the training data. This is exemplified when training an autoencoder on QCD jets to identify anomalous top jets, versus training the autoencoder on top jets to identify anomalous QCD jets. The underlying physics suggests that it should be easy to find large regions of phase space exclusively populated by QCD or top jets, therefore allowing for out-of-distribution detection in both directions. However, the autoencoder works well in the direction of anomalous top jets, while it does not work well in the direction of anomalous QCD jets. We refer to any anomaly detection technique that tags in both directions of higher and lower jet complexity, without modifications in the architecture and training, as symmetric.

A known issue in training EBMs is a potential collapse of the sampler, detected by a diverging negative energy and a collapse of the sampled images [49, 194]. To find a sweet spot between mode coverage and stability requires careful tuning of the LMC parameters, in addition to a regularization. We only encounter this failure when training on top jets, because the latent space undergoes drastic changes. To detect a collapse, we use several diagnostic tools. A proper training shows stable positive and negative energies, a fluctuating loss function close to zero, and smooth variations of the weights. In addition, we can directly look at the sampled images saving batches after a fixed number of iterations. The NAE training is carried over for 50 epochs or until a collapse of the sampler happens. Then, the best model is chosen by taking the iteration with the loss function closest to zero and with stable positive and negative energies.

We choose a three-dimensional latent space for our model, making it a sphere in three dimensions. We exploit this low dimensionality to visualize the development of the latent energy landscape. We employ an equirectangular projection as shown in Fig. 5.2. The x-axis and the y-axis give the longitude and the latitude on the sphere. To reduce the distortion around jets, the poles are chosen such that the center $(0, 0)$ is given by the region with most jets. By sampling points on the sphere and calculating the energy of the decoded jets we build the latent landscapes. In this landscape we show the path of latent LMCs and the position of encoded jets from both distribution.

In the upper panels of Fig. 5.2 we show a projection of the latent space after minimization of the MSE, like in the usual AE, but using a compact, spherical latent space. In the left panels we train on the simpler QCD background, which means that the latent space

LMC parameters	latent	input
λ	100	50
σ	10^{-2}	10^{-4}
# of steps	30	30
metropolis	✓	✓
annealing	–	✓
training parameters	**pre-AE**	**NAE**
learning rate	10^{-3}	10^{-5}
iterations	15k	40k
batch size	2048	128

Table 5.1: LMC and training parameters. The temperature is implicitly fixed by the noise and the step size as $T_x = 10^{-7}$ and $T_z \approx 10^{-6}$.

has a simple structure. The QCD jets are distributed widely over the low-energy region, while the anomalous top jets cluster slightly away from the QCD jets. The situation changes when we train on the more complex top jets, as shown in the right panels. The latent MSE or energy-landscape reflects this complex structure with many minima, and top jets spread over most of the sphere. After the NAE training, only the regions populated by training data have a low energy. The sampling procedure has shaped the decoder manifold to correctly reconstruct only training jet images. For both training directions, the Markov chains move from a uniform distribution to mostly cover the region with low energy, leading to an improved separation of the respective backgrounds and signals.

To see the difference in the two-directional training we can also look at the respective energy distributions. In the left panel of Fig. 5.3 we first see the result after training the NAE on QCD jets. The energy values for the background are peaked strongly, cut off below 4×10^{-5} and with a smooth tail towards larger energy values. The energy distribution for top jets is peaked at larger values, and again with an unstructured tail

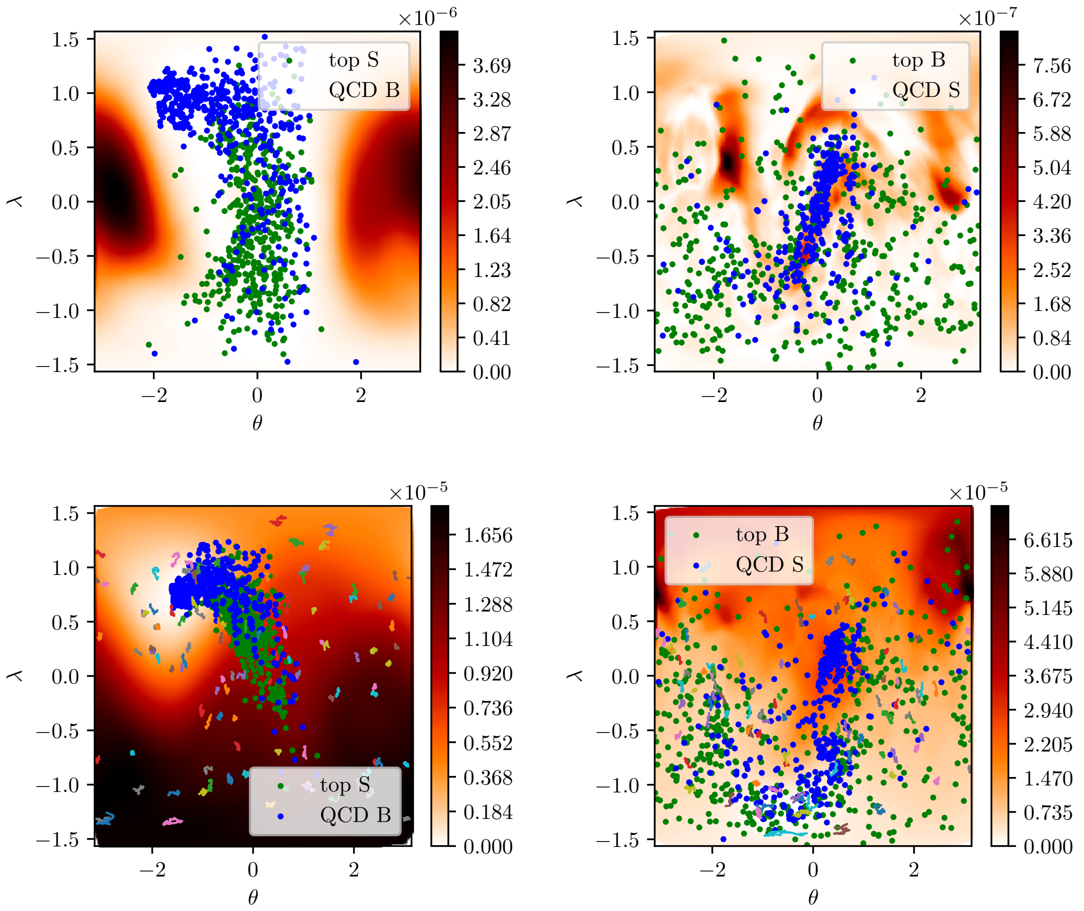


Figure 5.2: Equirectangular projection of the latent space after pre-training (upper) and after NAE training (lower). The x- and y-axis are the longitude and latitude on the sphere. We train on QCD jets (left) and on top jets (right). The background color indicates the energy over the latent space, the lines represent the path of the LMCs in the current iteration, and the points show the positions of jets from both samples.

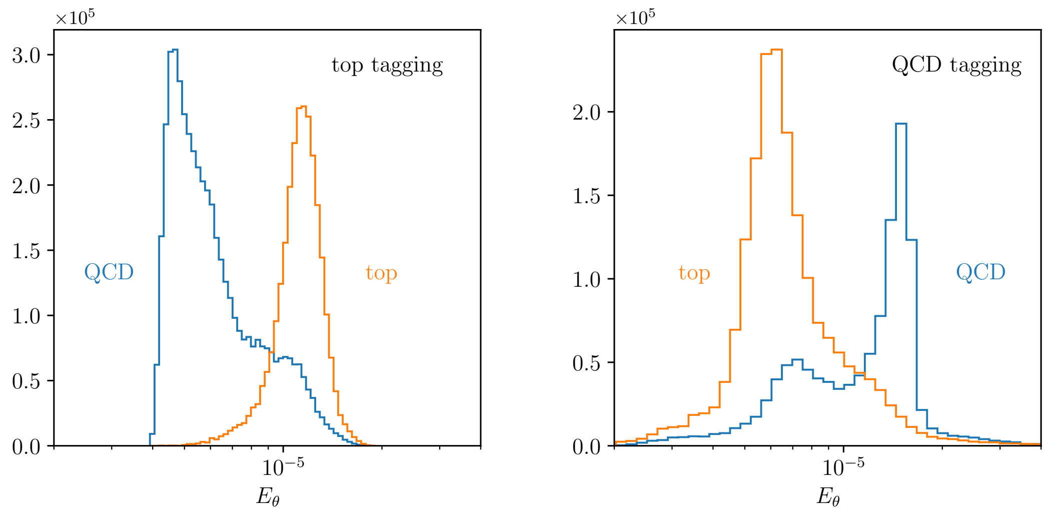
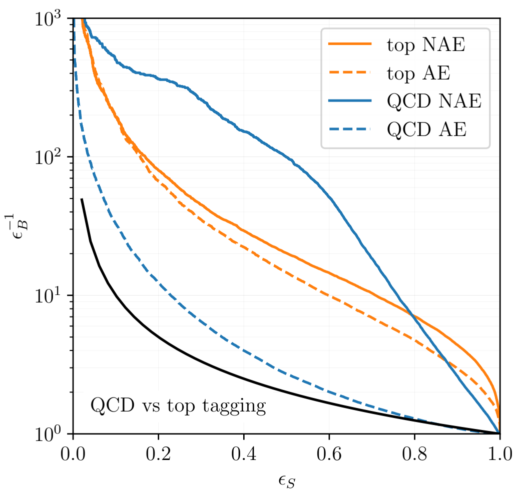


Figure 5.3: Distribution of the energy or MSE after training on QCD jets (left) and on top jets (right). We show the energy for QCD jets (blue) and top jets (orange) in both cases.

into the QCD region. We can then evaluate the performance of anomalous top tagging in terms of the ROC curve, the AUC score, and the inverse mistag at low efficiency ($\epsilon_s = 0.2$) in Fig. 5.4. This choice of working point is motivated by possible applications of autoencoders requiring significant background rejection. The orange ROC curves show how the performance increases after the additional AE training to the NAE training in the self-constructed latent-space geometry. The AUC value of 0.91 quoted in the corresponding table is above the AE setup and our earlier studies.

Next, we can see what happens when we train on top jets and search for the simpler QCD jets as an anomaly. In the right panel of Fig. 5.3 the background energy is much broader, with a significant tail also towards small energy values. The QCD distribution develops two distinct peaks, an expected peak in the tail of the top distribution and an additional peak under the top peak. The fact that the NAE manages to push the QCD jets towards larger energy values indicates that the NAE works beyond the compressibility ordering of the simple AE. However, the second peak shows that a fraction of QCD jets look just like top jets to the NAE. The ROC curves in Fig. 5.4 first confirm that training a regular AE to search for QCD jets in a top sample makes little sense, leading to an AUC value of 0.579. After the additional NAE training step we reach an ROC value of almost 0.9, close to the corresponding value for top tagging. However, the shape of the ROC curve does not exactly follow our expectations. We can start with large $\epsilon_S \to 1$ in the right panel of Fig. 5.3. Here the working point is in the small-energy tails of the signal and background distributions, and because of the tails in the top jet distribution the performance of the classification network starts poorly. Moving towards smaller ϵ_S the network performance drastically improves, until we pass the background peak, corresponding to $\epsilon_S \sim 0.6$. Below this value, the QCD tagging improves, again, but more slowly than the corresponding top tagging.

Altogether, we see in the right panels of Fig. 5.4 that the NAE combines competitive performance with symmetric tagging top and QCD tagging. In the easier direction of top tagging it beats the AE and DVAE benchmarks in spite of the non-optimized setup, and in the reverse direction of QCD tagging it provides competitive results for the first time.



Signal	NAE AUC	NAE $\epsilon_B^{-1}(\epsilon_S = 0.2)$	AE [139] AUC	DVAE [180] AUC
top (AE)	0.875	68	0.89	0.87
top (NAE)	0.91	80		
QCD (AE)	0.579	12	–	0.75
QCD (NAE)	0.89	350		

Figure 5.4: ROC curve for top (orange) and QCD (blue) tagging after AE pre-training (dashed), and after NAE training (solid). A random classifier corresponds to the solid black line. In the table we compare the performance of the NAE, and the pre-trained AE used here, to two studies in the literature.

5.2.2 Jets reconstruction and sampling

The NAE architecture allows us to check explicitly the reconstruction of different jets. Here, we show the average of two subsamples of QCD and top jets with their reconstruction. By comparing the input images and the reconstruction we get a better understanding of the main features learned by the network. These subsamples are shown in Fig. 5.5. The reconstruction of tops events as signal shows how the network is only able to reconstruct what's in the training distribution and therefore ignores additional prongs. In the inverse direction, the network detects all three prongs while also wrongly reconstructing QCD signal images. In the latter case the main contribution to the energy is coming from the intensity of the pixels rather than the location of the main prong.

Furthermore, we can explicitly look at sampled images and compare the average distribution to the expected one. The last two images of Fig. 5.5 show the average sampled distribution for QCD and top tagging. The averaging is performed on 1000 images sampled after training. In both cases the model distribution has converged and resembles the training background. A subset of the LMC samples is shown in Fig. 5.6.

5.2.3 QCD vs dark jets

After testing NAE on this benchmark process, we can move to a more difficult task, namely tagging two distinct kinds of dark jets with the same network. The signal datasets

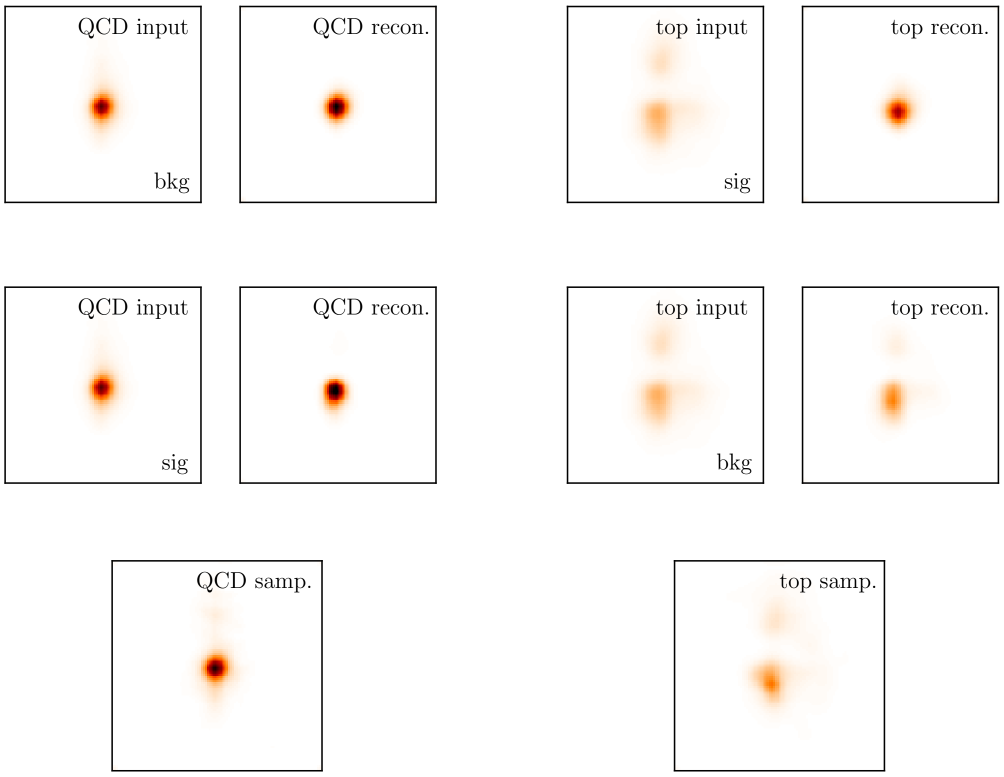


Figure 5.5: Average of various jet images. The first two rows account for the direct and the inverse tagging problem, and they are showing the average of 10k input and reconstructed images. The last row show an average of 1000 images sampled via LMC when training on the two different backgrounds.

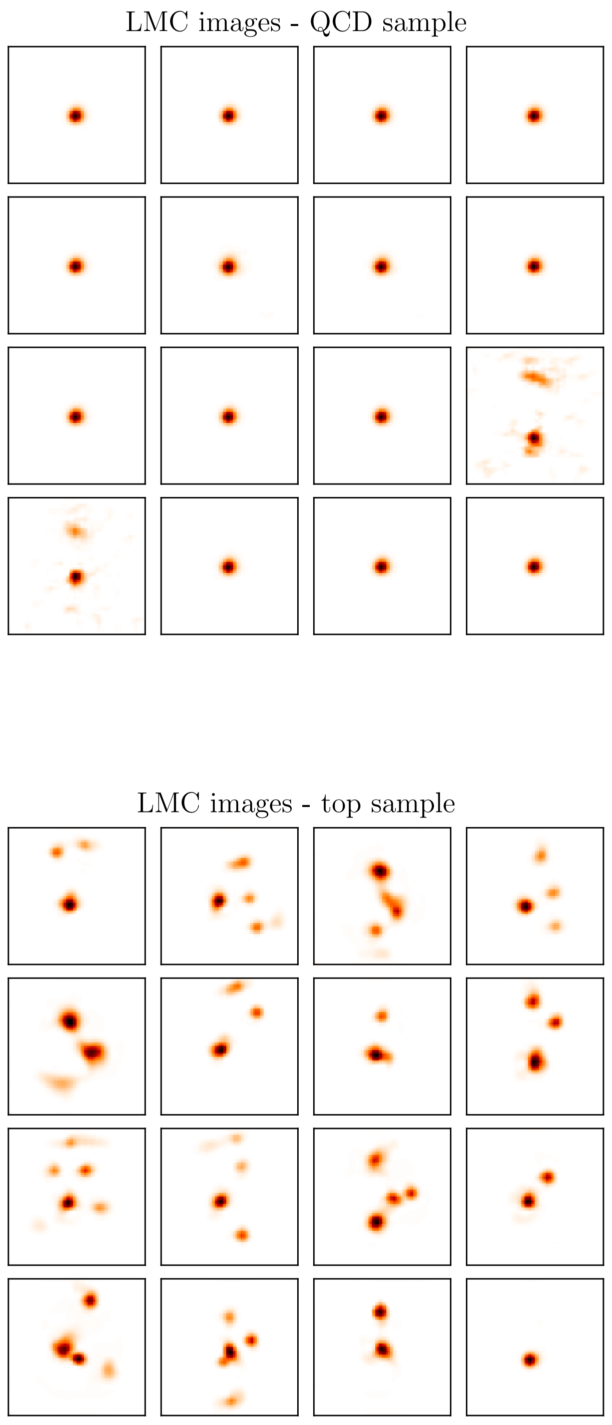


Figure 5.6: A subset of LMC samples for top (upper) and QCD (lower) tagging.

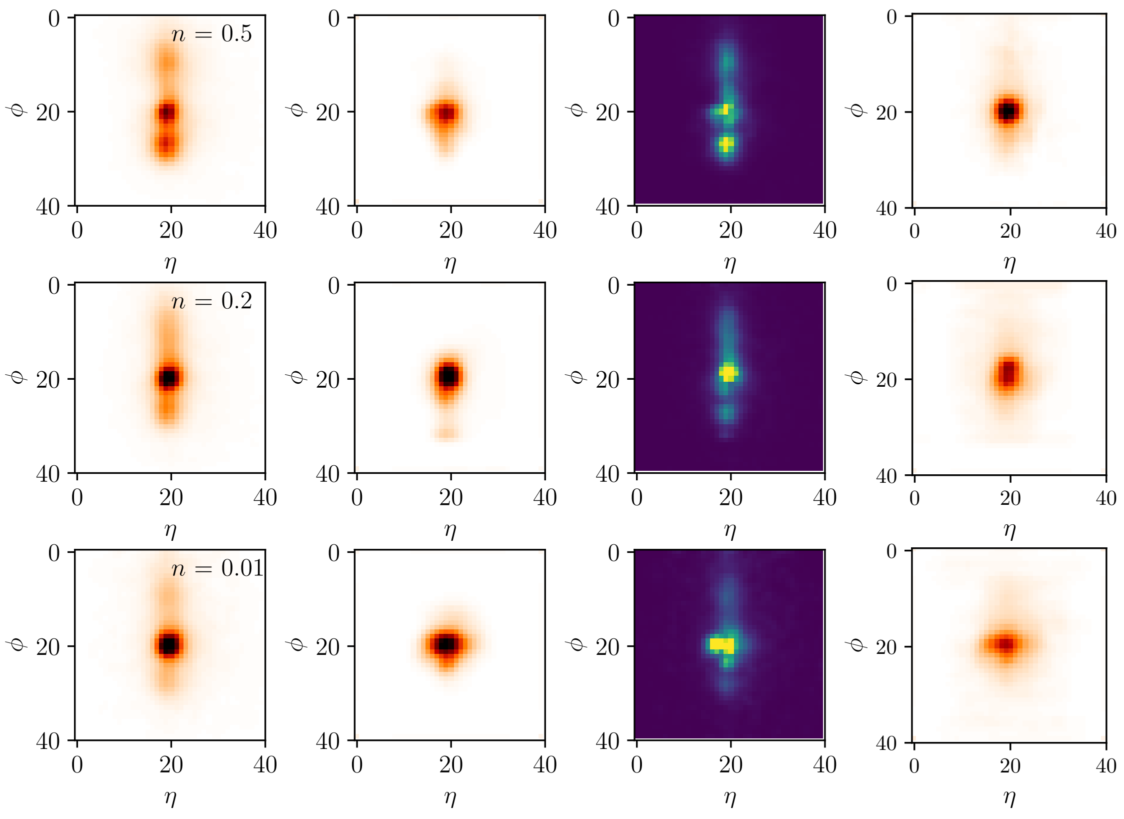


Figure 5.7: Average QCD jet images for the 1k most poorly reconstructed jets, from left to right: average input, average reconstruction, pixel-wise energy between the two, and average output of the negative energy sample used during training in the last iteration. The rows correspond to reweighting factors $n = 0.5, 0.2, 0.001$.

are the same as in Ref. [58].

To first illustrate the p_T-reweighting we select the most poorly reconstructed 1000 QCD images, according to their MSE or energy. In Fig. 5.7 we show the average of these images to the left, the average reconstruction in the second column, and the pixel-wise energy between the two in the third row. Reducing the remapping defined in Eq. (5.5) from $n = 0.5$ to $n = 0.01$ washes out the p_T-structures, so the input and especially the reconstructed images change from more structured jets to a simple, single-prong structure. For our two signal hypotheses this means that for large n the poorly reconstructed QCD images resemble the Heidelberg signal, leading to a more efficient signal extraction, while for small n the poorly reconstructed jet images resemble the Aachen dataset.

This difference in the jet reconstruction for different models can be explained by looking at the sampled distributions during training. The NAE-sampled average of the negative-energy jets in the last iteration is shown in the two right column of Fig. 5.7. At $n = 0.5$ the NAE sampling discards all secondary clusters and focuses on the main feature of the QCD jets, the single prong. During training, the loss function enhances the main feature by increasing the energy of everything around it in the latent and phase spaces. As a consequence, the initial background is lost after some epochs, but to keep the normalization of each jet the central prong is enhanced. As a result, the tagging of two-prongs structure like the Heidelberg jets is improved. Conversely, at $n = 0.01$ the reweighting enhances the secondary cluster, which cannot be discarded by the training

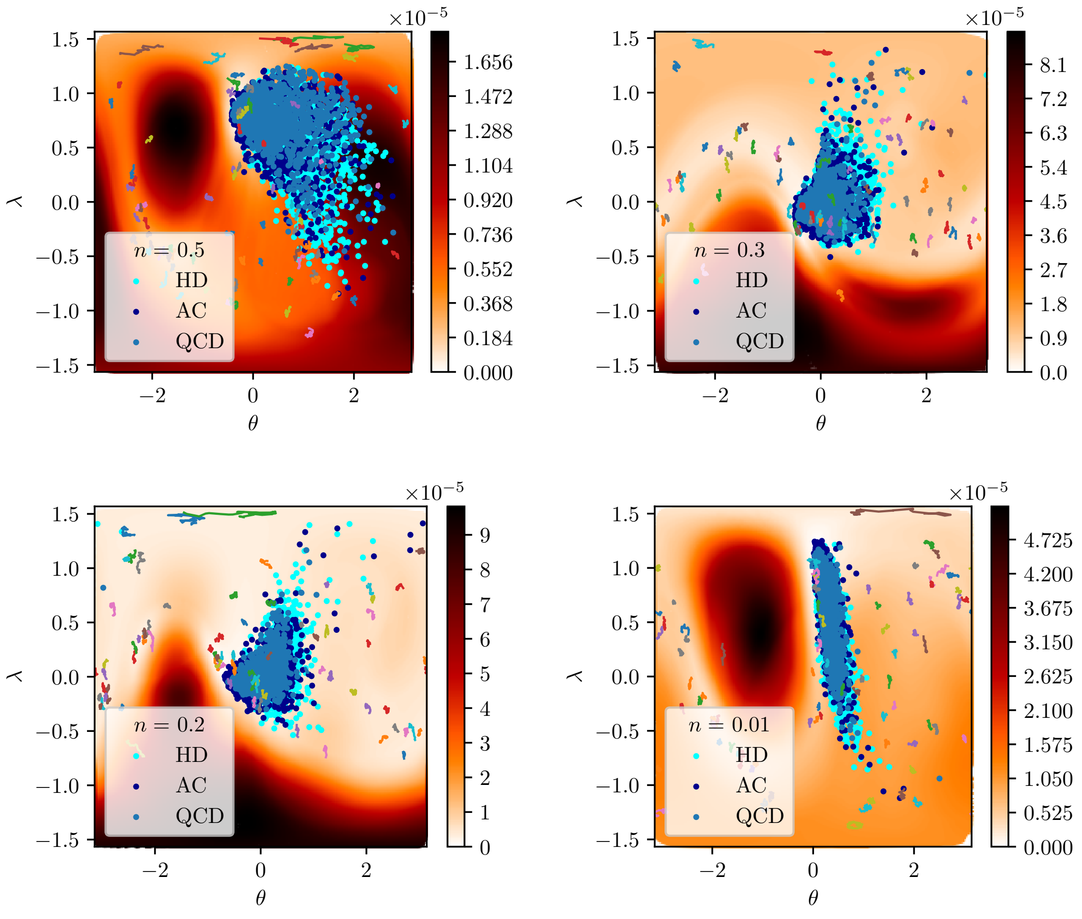
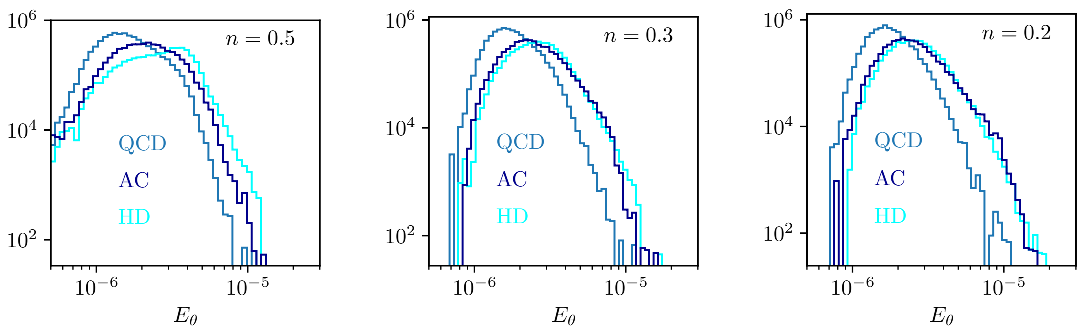
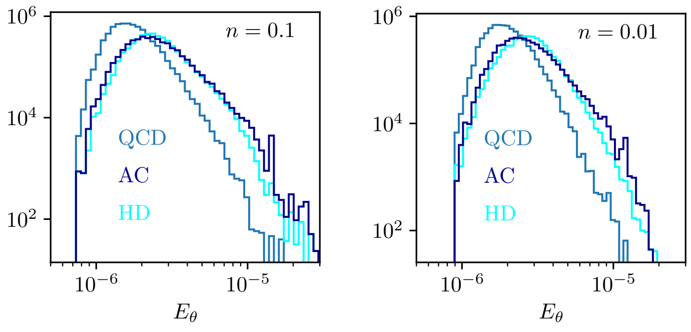


Figure 5.8: Equirectangular projection of latent spaces after NAE training on QCD jets to identify anomalous dark jets. We show four different p_T-reweightings n. The blue points represent a sub-sample of QCD training events.

anymore. As a result, in both initial iteration, and at equilibrium there is a residual background away from the central prong. This way, the NAE training increases the energy for Aachen jets, because the normalization forces the main prong to a lower value. These effects are inevitable when training likelihood-based models, a different preprocessing will change the density and, therefore, the anomaly score.

As for the top vs QCD tagging, we then show the latent space landscapes after NAE training in Fig. 5.8. For all p_T-reweightings the network identifies the least populated regions in the decoder manifold and increases the corresponding energy. As discussed above, a large $n = 0.5$ enhances the sensitivity to the more complex Heidelberg dataset, while the sparse Aachen dataset is hardly separated from the QCD jets. For small $n = 0.01$ a distinct region appears at large λ, where the Aachen signal extends beyond the QCD region. In between the two extremes, the latent landscape changes smoothly with the biggest change happening around $n = 0.2$. Around this point the training can oscillate between focusing on primary prongs or on secondary clusters, causing fluctuations in the performance. This transition in the p_T-reweighting is also the only case where the hyperparameters, and especially the temperatures, have a noteworthy effect on the network performance.





	QCD		HD		AC	
n	μ	σ	μ	σ	μ	σ
0.5	2.0	0.9	3.3	1.4	2.7	1.1
0.3	2.0	0.8	3.1	1.3	3.0	1.3
0.2	2.0	0.8	3.1	1.3	3.0	1.5
0.1	2.0	0.9	3.2	1.7	3.4	2.2
0.01	2.2	0.8	3.3	1.4	3.4	1.8

Figure 5.9: Distribution of the energy for QCD, Aachen, and Heidelberg datasets. Each panel corresponds to a different reweighting of the same datasets. The table shows the mean and the standard deviation for each distribution ($\times 10^{-6}$).

Once again focusing on the p_T-reweightings we show the energy distributions for the QCD training data and the two signals in Fig. 5.9. We see that unlike in our earlier study [58] the effect of the preprocessing on the whole distribution is limited. A shift in performance at low signal efficiency can be seen by varying n with the ordering between

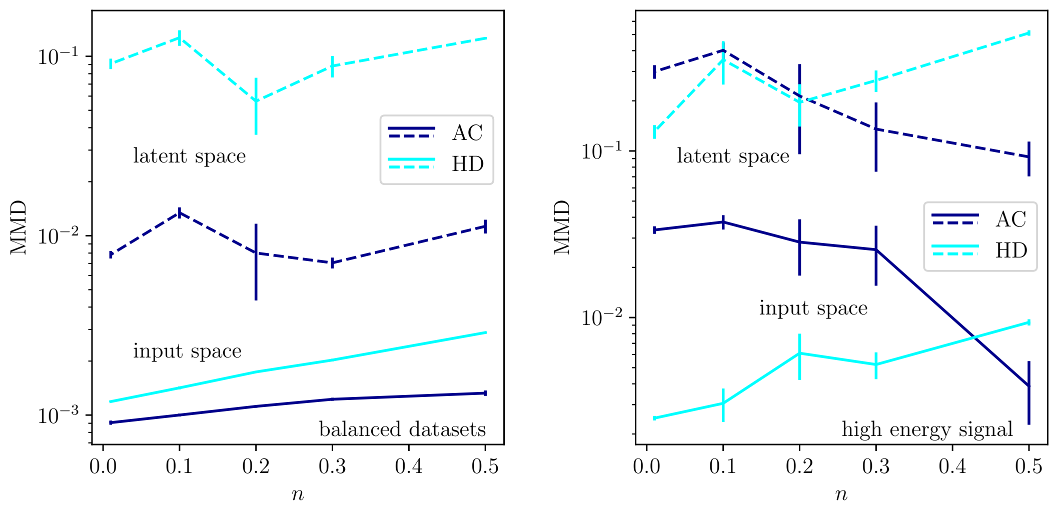


Figure 5.10: MMD in phase (solid) and latent (dashed) space between two random sample of QCD and signal jets (left), and between the QCD sample and the most poorly reconstructed signal jets (right). Unlike for the other figures, the remapping n increases from left to right.

the two dataset being switched around $n = 0.2 \dots 0.3$. The energy distribution of the Heidelberg dataset has a shifted main peak at $n = 0.5$ which is washed out by smaller reweighting factors, while the QCD distribution undergoes a slight shift and develops a longer high-energy tail. For the Aachen dataset, lowering n moves the mean away from the QCD background and at the same time increases the width of the distribution. These patterns will affect the ROC curves at low signal efficiency and large background suppression.

Once we understand how the p_T-reweighting changes the input distributions and the energy distributions for the QCD background and the dark jets signals, we can measure the difference between QCD jets and each of the two signals by computing the maximum mean discrepancy (MMD) [197] of sub-samples from these distributions. We show these MMD curves as a function of n in Fig. 5.10, for the full distributions of 20k QCD and signal jets and only considering the 1k most poorly reconstructed jets in the high-background-rejection target region. For the input distributions we see the same trend in both panels — small n benefits the tagging performance for the Aachen dataset and decreases for the Heidelberg dataset; increasing n improves the tagging performance for the Heidelberg dataset.

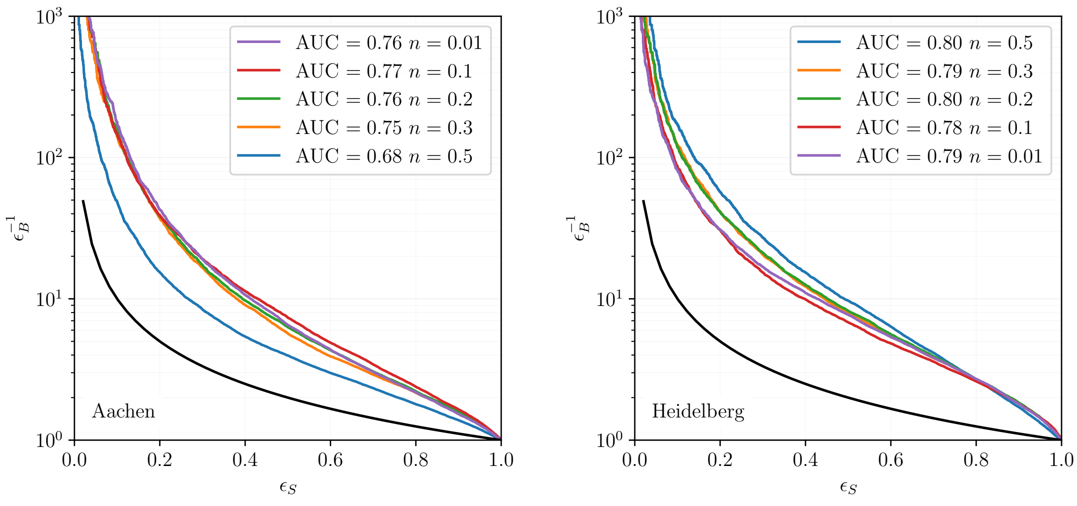
The more interesting question is if the input-space pattern can also be observed in the latent-space distributions. The idea behind this test is to construct an autoencoder with a choice of anomaly scores, either reconstruction-based in phase space or in the latent space [180]. Again in Fig. 5.10 we show the corresponding MMD values as dashed curves. For the complete samples as well as for the most poorly reconstructed jets the latent-space MMD behaves just like the phase-space MMD. This indicates that it should, if necessary, be possible to construct a latent-space anomaly score for the NAE.

Moving on to the performance of the NAE on dark jets, we show the ROC curves with different reweightings in Fig. 5.11. First, we see that the AUCs for the Aachen and Heidelberg datasets are roughly similar. For the sparse Aachen jets we already know that smaller values of n benefit the tagging performance, but we also see that for $n < 0.3$ the AUC reaches values above 0.72, and for $n = 0.2 \dots 0.01$ the performance essentially plateaus at a high level. In contrast, for the Heidelberg signal we expect a better tagging performance around $\epsilon_S \sim 0.2$ for larger n-values.

From Fig.5.9 we know that the different reweightings mostly change the ordering of the two signal tails at high energies and leave the bulks of the distributions unchanged. The corresponding ROC curves in Fig. 5.11 confirm that the remaining n-dependence is connected to a behavioral change in the model in the region $n \sim 0.2$. While the choice $n = 0.2$ is not optimal for each of the signals, it can be used as a working compromise between sparse dark jets and dark jets related to a mass drop.

5.3 AnomalyCLR & DarkCLR

In this section we describe the AnomalyCLR and the DarkCLR methods. We discussed in Sec. 3.2 the contrastive learning of representations [101] technique used to construct highly-expressive representations of data for use in downstream tasks, which for us is anomaly detection. The advantage of this, from the collider physics perspective, is that the technique can be run directly on experimental data rather than on simulation. This allows the network to learn non-trivial correlations and tails in data that might not be well-modelled in simulations.



Data		n 0.5	0.3	0.2	0.1	0.01
Heidelberg	AUC	0.795 (5)	0.796 (5)	0.789 (8)	0.78 (1)	0.790 (5)
	$\epsilon_B^{-1}(\epsilon_S = 0.2)$	62 (3)	42 (5)	42 (4)	28 (4)	30 (1)
Aachen	AUC	0.68 (1)	0.746 (5)	0.75 (1)	0.767 (5)	0.755 (5)
	$\epsilon_B^{-1}(\epsilon_S = 0.2)$	15 (1)	38 (3)	33 (7)	41 (2)	41 (1)

Figure 5.11: ROC curve for dark jets tagging with different reweightings n, shown for the Aachen signal (left) and the Heidelberg signal (right). A random classifier corresponds to the solid black line. The table is based on the same information and shows the mean and the standard deviation of five different runs.

5.3.1 Contrastive learning for anomaly detection

While contrastive learning has been shown to be very useful in generating representations for downstream classification tasks [198], there is a potential issue when using this approach for downstream anomaly detection tasks. For the classification task, for example in [198], the function $f(\cdot)$ is optimised on data from both the background and signal classes, despite not using their truth-labels explicitly in the optimisation. Through the contrastive learning this allows the function to encode non-trivial features of both the background and signal data in the representations. When using contrastive learning for a downstream anomaly detection task however, the function f is optimised on just the background data (or at least a significantly background-dominated dataset). This means that the representation learned by the function $f(\cdot)$ will focus solely on features relevant for the background data. This could mean that anomalous data is not out-of-distribution and so may not lead to competitive performance in downstream anomaly detection tasks. This will become evident when we look at the results in Sec. 4.3.2. To remedy this we introduce AnomalyCLR, a modified approach to contrastive learning for anomaly detection in particle physics. At the core of this approach is the introduction of 'anomaly-augmentations', such that we now have two categories for augmentations:

1. Physical augmentations
   These are augmentations of the data that we would like the mapping to be invariant to.
2. Anomaly-augmentations
   These are unphysical augmentations of the data that are supposed to mimic potential anomalies, we want the representations to be highly discriminative towards these augmentations.

We add a third pseudo-label:

3. Anomaly-pair labels
   These labels match each data point in the sample to an anomaly-augmented version of itself.

The advantage of anomaly-augmentations is that we can increase the sensitivity of the anomaly detection tools to anomalies using just the background data, potentially the data directly measured at colliders. This keeps the approach in line with the original data-driven CLR idea. We can then define the anomaly-augmented contrastive loss function as

$$\mathcal{L}_{\text{AnomCLR}} = -\log \frac{e^{\left[s(z_i,z_i')-s(z_i,z_i^*)\right]/\tau}}{\sum_{j\neq i\in\text{batch}}\left[e^{s(z_i,z_j)/\tau}+e^{s(z_i,z_j')/\tau}\right]} \,, \tag{5.9}$$

where we denote the representations of the anomaly-augmented events by z^*, and so the anomaly-pair is defined as $\{(x_i, x_i^*)\}$. Note that the anomaly-augmentations only enter in the numerator of Eq. (5.9), and without these the loss function becomes the regular contrastive loss function. Introducing the anomaly-pairs we expose the network to data features that are outside of the background distribution. The CLR portion of the loss function still optimises for alignment and uniformity, however this uniformity is now disrupted by the anomaly-pair term. As a result the background data will not be uniformly distributed in the representation space, with some regions encoding features of the anomaly-augmented data. This means that anomalous data with features similar to those generated by the anomaly-augmentations should be out-of-distribution in this representation space.

We did some minor testing on alternative forms of this loss function, for example including the anomaly-augmentations in the denominator of the loss function with the negative-pairs. However since the anomaly-augmentations compute distances between a data point and its augmented counter-part, and not between other data points (i.e. $i \neq j$), it is more intuitive to include this term in the numerator. The denominator in Eq. (5.9) is used to encode features in the representation space that discriminate between the different data points used during training, which for anomaly detection is the background data. This is not necessary for anomaly detection, and the anomaly-pairs should provide the representations with all the discriminative power they need, so we experimented with removing the denominator in Eq. (5.9) altogether, and found that this is sufficient. In this case the loss function is written as

$$\mathcal{L}^{+}_{\text{AnomCLR}} = -\log e^{\left[s(z_i,z_i')-s(z_i,z_i^*)\right]/\tau} = \frac{s(z_i,z_i^*)-s(z_i,z_i')}{\tau} \,, \tag{5.10}$$

where the plus sign in $\mathcal{L}^{+}_{\text{AnomCLR}}$ indicates that only positive-pairs are used. This results in a much less computationally expensive loss function, since we no longer need to compute pair-wise correlations between each entry in a batch the complexity scales as N_{batch} rather than N^2_{batch}. We also remove the dependence on τ in $\mathcal{L}^{+}_{\text{AnomCLR}}$, since there is no longer a trade-off between positive- and negative-pairs. We could of course introduce a term to control the trade-off between the physical and anomaly-augmentation terms, but we do not explore that here. In our results we will compare the performance of both loss functions. The number of augmentations is theoretically unlimited, however including a large number of scenarios can incur unstable optimization of the loss especially for contradicting transformations. This problem can be tackled with a larger batch size, to get a better average estimate of the loss, and with a larger representation space.

5.3.2 Applications to events and semi-visible jets

The application of AnomalyCLR to different physical scenarios requires an understanding of the data and the physics in order to construct the physical and anomaly-augmentations. For the event-level dataset discussed in Sec. 5.1 we consider three physical augmentations to the data:

1. Azimuthal rotations
   The whole final state is rotated by an angle ϕ randomly sampled from $[0, 2\pi]$.
2. $\eta - \phi$ smearing
   The (η, ϕ) coordinate of every object in the event is resampled according from a Normal distribution centred on the original coordinate and with a variance inversely proportional to the p_T, i.e. $\eta' \sim \mathcal{N}(\eta, \sigma(p_T))$ and $\phi' \sim \mathcal{N}(\phi, \sigma(p_T))$.
3. Energy smearing
   The p_T of every object in the event is re-sampled according to $p'_T \sim \mathcal{N}(p_T, f(p_T))$ with $f(p_T)$ determining the strength of the smearing.

These augmentations reflect both the symmetries in the data and the experimental resolution of the detector. Detectors are imperfect, especially in measuring jet energies, and we encode this in the representations of the data through the energy-smearing augmentation. Here we re-sample the jet p_T's as $p'_T \sim \mathcal{N}(p_T, f(p_T))$, where $f(p_T) = \sqrt{0.052 p_T^2 + 1.502 p_T}$ is the energy smearing applied by Delphes (the p_T's are normalised by 1GeV). If not explicitly mentioned, we always assume units of GeV for energy. For the anomaly-augmentation we consider some very simple scenarios:

1. Multiplicity shift, $x'_i = m(x_i)$
   For each event $m(\cdot)$ adds a random number of electrons, muons, and jets to the event. The number is chosen randomly within the limits $(n_e, 4{-}n_e)$, $(n_\mu, 4{-}n_\mu)$, and $(n_j, 10{-}n_j)$ for electrons, muons, and jet, respectively. The azimuthal angle and pseudo-rapidities are also chosen randomly within the limits allowed, and the p_T for each object is chosen as a random fraction of the maximum p_T in the event. Once the objects have been added, the MET of the event is recalculated and updated.
2. Multiplicity shift, keeping MET and the total p_T constant, , $x'_i = \overline{m}(x_i)$
   This is similar to the above augmentation, but now $\overline{m}(\cdot)$ generates the extra objects by splitting the existing objects and smearing the $\eta - \phi$ coordinates using the function used in the physical augmentations above.
3. p_T and MET shifts, $x'_i = s_{p_T}(x_i)$
   Here $s_{p_T}(\cdot)$ shifts the p_T's in the event by the same random factor. We randomly

choose whether we shift just the MET, just the reconstructed object p_T's, or both. And we ensure that the the trigger selection is not spoiled by these shifts.

With the physical augmentations we apply all of them simultaneously, whereas for the anomaly-augmentations we apply just one augmentation to each event. The augmentation that is applied is selected randomly and uniformly. We do not apply both a physical augmentation and an anomaly-augmentation to the events in $s(z_i, z_i^*)$, since this would conflict with the optimisation goal of the $s(z_i, z_i')$ term. It would also be possible to have an anomaly-augmentation that removes objects from the event, however this effect is already captured by the augmentation that adds objects to the event. Many of the events in the background have the minimal multiplicity allowed by the applied cuts, so the effect of an anomaly-pair with a low multiplicity background event and the same event augmented to have more objects is the exact same as the effect of an anomaly-pair with a high-multiplicity background event augmented to have less objects. This is because of the symmetry in the distance function $s(z_i, z_i^*)$. So the anomaly-augmentations here are as general as can be, and do not target any specific new physics scenario, therefore the technique should be model-agnostic. More precisely, the anomalous transformations democratically introduce a modification given the background data and its format. In our implementation there are no explicit assumptions on the allowed signals, both multiplicity shifts, $m(\cdot)$ and $\bar{m}(\cdot)$, uniformly sample the number of additional reconstructed objects and differ in the treatment of the kinematics. In the first case, the new objects take a fraction, uniformly sampled as well, of the maximum p_T, while in the second case the total MET and the p_T are left constant. The third augmentation shifts MET and/or p_T uniformly sampling a scaling factor, given the original values. Here, we fixed a window corresponding to five times the original momentum. This window has not been fine tuned and, given the wide range of kinematics in the training events, this window covers an extremely large phase-space region. Given the generality of these transformations, the representations comply with our anomaly detection downstream task. For the general application of this method, it is important a careful study of the augmentation technique and their implementations to avoid the usage of inconsistent data.

In DarkCLR we apply the CLR framework to the semi-visible jets scenario presented in Sec. 5.1. We start with the positive augmentations. These are easy to implement approximate symmetries of a jet:

- **Rotations:** We rotate each jet in $\eta - \phi$ by an angle which is chosen randomly between $[0, 2\pi]$. Note that the angle is chosen randomly for each jet, i.e., each constituent inside a jet is rotated by the same angle.
- **Translations:** We shift each constituent in the $\eta - \phi$ plane by randomly choosing a shift in a window with size given by the distance between the two furthest constituents.

After applying these two transformations to the original jet x_i, we obtain the augmented version x_i' and the positive pair $\{x_i, x_i'\}$.

Semi-visible jets, as discussed earlier, have fewer constituents than QCD jets. Therefore, we consider dropping constituents as an **anomaly augmentation**. The transformation is implemented as follows: We drop each component of the jet with a fixed probability p_{drop}, and the p_T of the augmented jet is rescaled to match the original p_T. The latter step ensures that the augmented jets fulfill the selection cuts applied in the generation process.

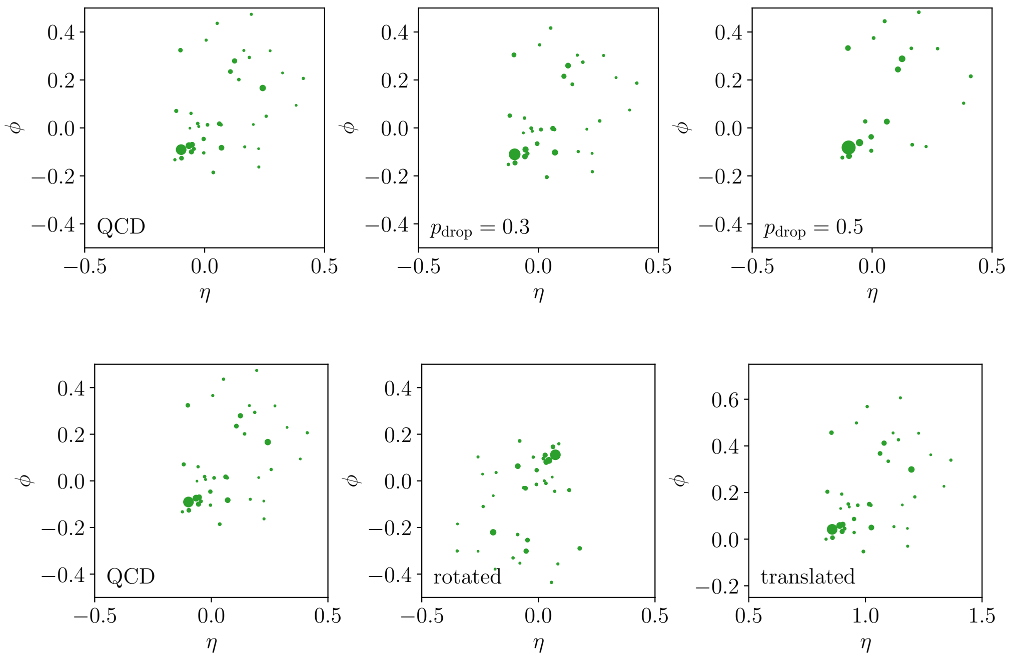


Figure 5.12: (top) Example of positive augmentations on a QCD jet. The original QCD jet is rotated in $\eta - \phi$ the middle panel and translated in $\eta - \phi$ in the right panel. (bottom) Example of an anomalous transformation on a QCD jet. The left panel shows the original background jet while the middle and right panels show the same jet after applying the augmentations with $p_{\text{drop}} = 0.3$ and $p_{\text{drop}} = 0.5$ respectively.

Fig. 5.12 shows an example positive transformation of a QCD jet used during training, together with an anomalous one with $p_{\text{drop}} = 0.3$ and $p_{\text{drop}} = 0.5$.

5.3.3 Network architecture

A clear difference stems from the event-based datasets compared to jets. The collider event data being used has a well-defined structure:

- MET: one entry with (p_T, η, ϕ)
- Electrons: four entries, each with (p_T, η, ϕ)
- Muons: four entries, each with (p_T, η, ϕ)
- Jets: ten entries, each with (p_T, η, ϕ).

The multiplicity is typically much less than the maximum allowed, so the data for a single collider event can have many zeros. A transformer netowrk allows us to avoid this by having a permutation-invariant and variable length input format. Because the data is now processed in a permutation-invariant way, the information on which entry corresponds to which object (MET, electron, muon, or jet) is lost. We reinstate this information by adding a one-hot encoded ID vector to (p_T, η, ϕ), with a 1 indicating the correct ID. This means that each reconstructed object is now represented by a 7D vector. Before passing the kinematic data to the transformer we do some very minor preprocessing to make sure that the numbers the networks see are $\mathcal{O}(1)$. Specifically, we divide all MET and p_T values by the average p_T of all objects (electrons, muons, jets) in the background dataset, we do not shift the values to be centred on zero because the distribution is highly peaked at zero and we want the preprocessed data to have the same sparsity as the original data. We then divide all η and ϕ values by 4 and π, respectively. When training the AutoEncoder networks discussed in the next section we use the same preprocessing of the data, this ensures that any difference in the results can be attributed to AnomalyCLR. On the contrary, the jet dataset has no internal structure and the full permutation invariance from the transfomer is desired.

We describe the network architecture following the schematic in Fig. 5.13. Notice that the indexed variable x refers to the objects inside the jet or event and not to one instance of the training data.

As the first step of the CLR network, an embedding layer maps the set of constituents $\{(p_{Ti}, \eta_i, \phi_i)\}_{i=1}^{N_c}$ to a larger vector with d_r dimensions. The number of features inside the jet/event has a fixed maximum size of N_c. This selection includes the entire event and the entirety of the QCD jets in most cases, while ignoring the softest components if the number of constituents is larger than N_c. The embedded constituents are then passed through a sequence of transformer encoder blocks. A block consist of a multi-head self-attention layer followed by a feed-forward network. A single-head self-attention operation transforms the input set by considering all the correlations between the constituents. The operations inside the self-attention block are described in Sec. 3.3. The output of the last transformer block provides an encoding of dimension (N_c, d_z). As a crucial next step, the output is summed over N_c to induce permutation symmetry between the features. Finally, the output is passed to a fully connected head network. The output of the head network then serves as the representation and input to the contrastive loss function of Eq. (5.10).

If a jet has less than N_c constituents, these are zero-padded. We ensure that this does not affect the transformer by masking the zero p_T entries. The masking procedure ensures that the attention weights from zero-padded constituents are zeros by adding minus infinity to the attention weight before normalization. Additionally, the contributions from the masked particles are also ignored in the final aggregation over N_c [198].

The output of this head network is what is passed to the loss function. We list the hyper-parameters used in training the networks in Appendix B. The representation used in the anomaly detection task is taken either from the output of the transformer network or the output of the head z.

5.3.4 Anomaly scores

Here we describe the anomaly scores we studied for the new representations. In AnomalyCLR we focused on the CLR loss function and used a standard autoencoder to test the representations. Indeed, it worked well because these benchmark have larger multiplicity. The semi-visible jets studied in DarkCLR provide a case study where we have more control on the phenomenology. In this case we studied the structure of the representations with an anomaly score defined directly in CLR space and, additionally, a NAE as better density estimator. In the following, we present the details of the three anomaly scores.

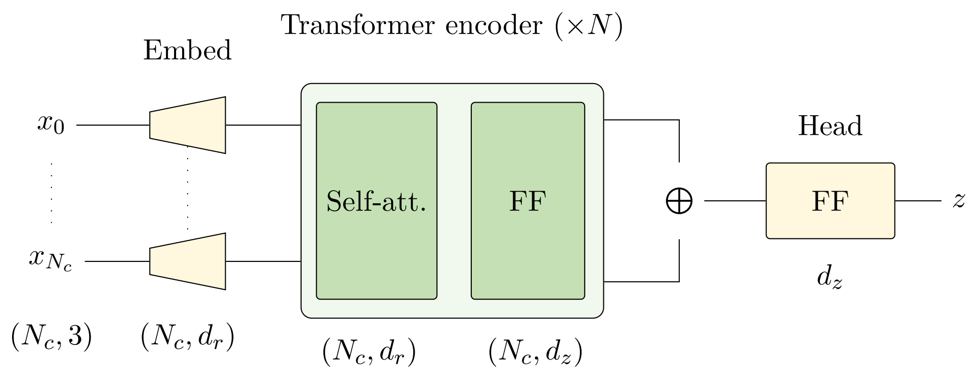


Figure 5.13: Schematic of the network architecture. The shape of the input vector, excluding the batch dimension, are shown after each step.

AE The encoder and decoder networks used in AnomalyCLR have 5 feed forward layers each with 256, 128, 64, 32, and 16 neurons, connected by a 5-dimensional bottleneck. The activation function between layers is a LeakyReLU with default slope. The decoder is a mirrored version of the encoder. We don't apply regularization techniques during training. The training is performed using Adam optimiser with learning rate 0.001 for 100 epochs, the batch size is 4096, and the number of SM events used is 10^6. Note that we have not optimised the AutoEncoder architecture, simply choosing the same architecture used in [77]. Instead we have only ensured that they are trained to convergence and that the training is stable. The AutoEncoder is trained on both the representations obtained from contrastive learning and the raw data. In the case of the raw data we apply the same preprocessing to the data as is applied to the data in the contrastive learning network. In this way we ensure that any differences in the anomaly detection performance can be attributed to the contrastive learning methods. As presented in Sec. 3.3 the MSE will be the final anomaly score.

CLR Here we focus more on understanding the CLR representations. First, we show in Appendix A that the representation before the head network encodes useful information for the discrimination between background and signal. In particular, the representations perform better than the constituents-level on a simple linear classifier test (LCT). However, we find that the output of the head network performs better on a cut-based analysis on a very simple quantity, and we use this representation for the evaluation of the anomaly scores. We first note that one way to reduce the loss is to simply increase the length of the vector so that jets with different properties are separated in the non-normalized space and close to each other after projection. Therefore, we expect the norm of the representation vector to be a discriminative scalar quantity and propose it as a CLR-based anomaly score that can show the effect of the DarkCLR pretraining. Namely:

$$s_{\text{CLR}} = ||z||_{L_2}, \qquad z \in \mathbb{R}^{d_z}. \tag{5.11}$$

Before using this anomaly score, a small modification is needed. Since our loss Eq. (5.10) is norm-free, the ordering between background and signal norms is not a priori fixed. This ambiguity, which can spoil applications in anomaly detection, is resolved by introducing a regularization term which penalizes background representations with large norms. This ensures that anomaly detection associates high norm with outlier data. The implementation is done by adding to the loss function the L_2 norm of the representations of the background batch. We find empirically that this new term does not affect the similarity, and therefore the loss, of the training. By definition s_{CLR} has no access to angular information which should provide additional discriminative information. We include this in the following anomaly score that takes as input the full high-dimensional vector.

NAE The second anomaly score we consider in DarkCLR is the reconstruction error of a normalized autoencoder. We remind that a NAE shares the same structure of a standard AE with the added robustness of maximum likelihood estimate training. The strategy for the analysis of the DarkCLR representations consist of obtaining the latent representations via the encoding function f defined in Sec. 5.3 and pass them to the NAE. The energy function is then used as anomaly score which is approximately invariant under the physical transformations used during the CLR training. Since the autoencoder is trained in a second step, DarkCLR can be seen as a pre-training procedure which exploits known invariants and the anomalous augmentation to provide better representations where we run a density estimation downstream task.

The following description of the NAE methodology differs from the one presented in Sec. 5.2 and therefore shortly summarized. We assume to train on representations z sampled from the latent space distribution $p_Z(z)$ induced by the training data,

$$z = f(x) \qquad \text{where} \qquad x \sim p_{\text{data}}(x)\,. \tag{5.12}$$

In the NAE we assume a Boltzmann underlying distribution p_θ with energy E_θ:

$$p_\theta(z) = \frac{e^{-E_\theta(z)}}{\Omega}, \qquad E_\theta(z) = ||z - z'||_2, \tag{5.13}$$

where θ are the trainable parameters of the network, and z' is the reconstructed representation.

Performing MLE on the probability distribution translates to minimizing the sum of the reconstruction error and the normalization factor Ω. However, computing Ω becomes easily intractable for high-dimensional spaces, so we do not explicitly minimize this quantity. Instead, we rewrite the gradient of the loss function in a computationally feasible manner as:

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{z\sim p_Z}[\nabla_\theta E_\theta(z)] - \mathbb{E}_{z\sim p_\theta}[\nabla_\theta E_\theta(z)]. \tag{5.14}$$

We obtain samples from p_θ using LMCs. An LMC process follows the equation:

$$z_{t+1} = z_t - \lambda \nabla_z \log p_\theta(z) + \sigma\epsilon \qquad \epsilon \sim \mathcal{N}(0,1), \tag{5.15}$$

and does not require an estimate of the integral due to the independence of the latter from the input z.

In particular, we utilize the contrastive divergence [199] Markov chain Monte Carlo scheme. Given a transition kernel T_θ for the data distribution p_Z, the following combination of Kullback-Leibler divergences has a zero only for $p_\theta(z) = p_Z(z)$ [200]:

$$\mathrm{KL}(p_Z||p_\theta) - \mathrm{KL}(T^t_\theta(p_Z)||p_\theta)\,, \tag{5.16}$$

Therefore, we can run short Langevin Markov Chains with steps t, which define the transition kernel T^t_θ, and estimate the gradients of Eq. 5.14 as:

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{z\sim p_Z}[\nabla_\theta E_\theta(z)] - \mathbb{E}_{z\sim T^t_\theta p_Z}[\nabla_\theta E_\theta(z)]. \tag{5.17}$$

Note that Eq. 5.17 ignores an additional term as pointed out in [199]. We find that this approximation does not affect the convergence of our model and therefore we use the base CD loss.

The procedure defined above stabilizes the training and corrects for the mismodeling of the density estimate introduced by the mere minimization of the reconstruction error. The epoch with the energy difference closest to zero defines the best loss, and we select the corresponding model for evaluation. Before turning on the regularization term, we pre-train the autoencoder for 200 epochs then continue training according to Eq. 5.14 for another 100 epochs. The architecture of the encoder network is a simple feed-forward network with five layers with neurons from 128 to 8 in powers of two and a three-dimensional bottleneck. The decoder mimics the encoder network, this time up-sampling from 8 to 128 dimensions in powers of two.

5.3.5 AnomalyCLR results

In this section we present some results using the different techniques discussed in the preceding sections. The results here are three-fold; we first compare the different methods based on anomaly detection performance, we then study the effects of the different anomaly-augmentations on the AnomalyCLR performance, and lastly we look at the effect of the representation dimension on the performance.

Comparison of methods We compare the methods using the ROC curves, the significance improvement (SI) curves, and the AUC. The baseline we compare to is the AutoEncoder trained on raw kinematic data. We present results using the CLR method without anomaly-augmentations ($\mathcal{L}_{\mathrm{CLR}}$), and the CLR method with anomaly-augmentations (both $\mathcal{L}_{\mathrm{AnomCLR}}$ and $\mathcal{L}^+_{\mathrm{AnomCLR}}$). So we have 4 methods in total to compare. For all results on the raw data we have trained five AutoEncoder networks and taken the central value and the error estimation from the mean and standard deviation of the results. For the CLR methods we also aggregate over five different CLR runs, where for each run we train a different transformer network and three different AutoEncoders. We then take the central value and the standard deviation for the error estimate. The CLR representations have a dimension of 160 and where anomaly-augmentations are used we have used them all as outlined in Sec. 5.3.2. In Fig. 5.14 we present AnomalyCLR results using $\mathcal{L}^+_{\mathrm{AnomCLR}}$ and see that it leads to significant improvements over the raw data representations, not only in the AUC but also at all signal efficiencies. In the Significance Improvement (SI) curves we also see large improvements, with the SI being between $\sim 3.5-4$ for $A \to 4l$ and h^+. We can see from Tab. 5.2 that the $\mathcal{L}^+_{\mathrm{AnomCLR}}$ loss function is clearly advantageous over $\mathcal{L}_{\mathrm{AnomCLR}}$, beating it on all signals with the exception of $A \to 4l$, where $\mathcal{L}_{\mathrm{AnomCLR}}$ achieves better performance at $\epsilon_s = 0.3$. A point of interest here is that the AutoEncoder on raw data outperforms the AutoEncoder on the CLR representations

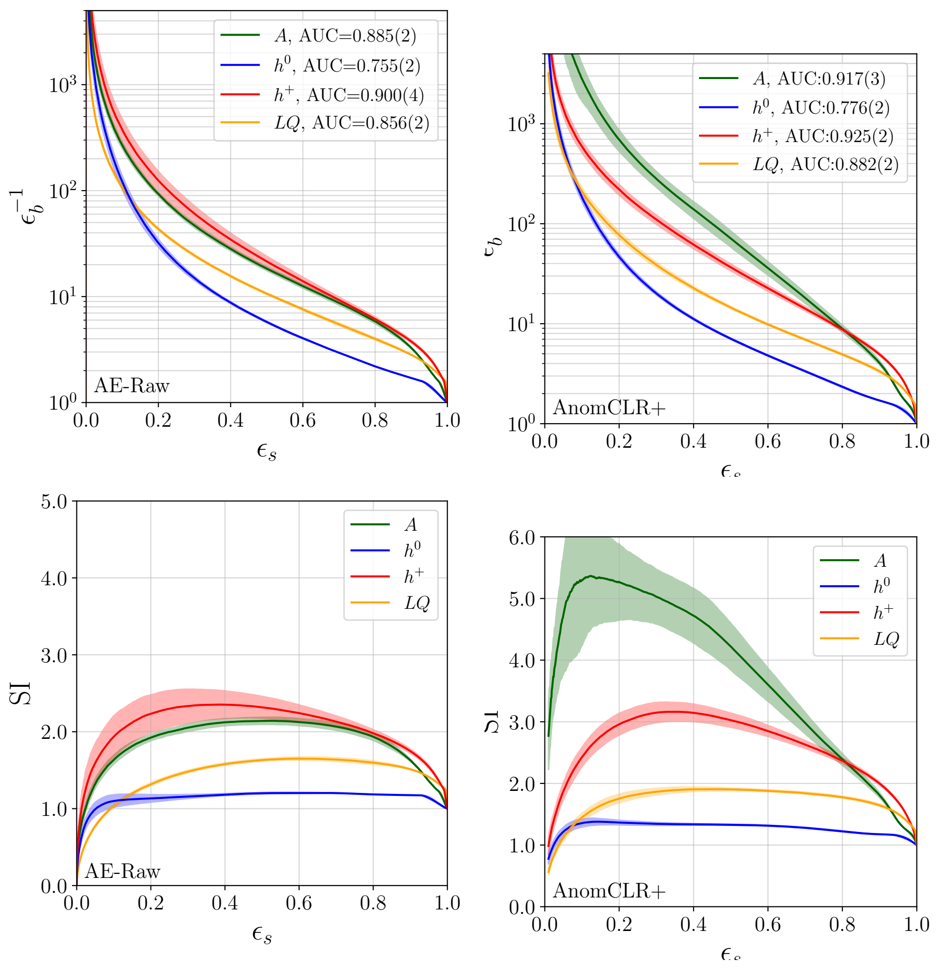


Figure 5.14: Comparison between the AE on raw data and the AE on the CLR representations trained with the $\mathcal{L}^{+}_{\text{AnomCLR}}$ loss function.

in most cases. This is likely due to the fact that traditional CLR optimises for uniformity, and since it is trained on background only, the mapping is not optimised to separate SM-like background events from any event which may look different to that. The benefit of anomaly-augmentations here is strikingly clear.

The effect of anomaly-augmentations We now want to study how the addition of the individual anomaly-augmentations affects the anomaly detection performance. For this we use just $\mathcal{L}^{+}_{\text{AnomCLR}}$, however we expect the results with $\mathcal{L}_{\text{AnomCLR}}$ to be similar. We use a representation dimension of 160 and obtain the error estimate on the runs with a combination of different CLR and AutoEncoder trainings. We train five different CLR models, and then train three separate AutoEncoders on each of these models, and take the average and standard deviation to obtain the error. We can see from Fig. 5.15 that the affect of the augmentations together results in more or less the best overall performance. One thing we noticed is that it can be difficult to determine from the affect of individual augmentations, or subgroups of them, what the performance of all of them together will be. For example, in most cases if we take just the $m(x)$ augmentation, i.e. the multiplicity augmentation that simply adds reconstructed objects, we see that it alone decreases performance below baseline for two out of four signals. However when used in combination with the others it either increases or has little effect on the performance.

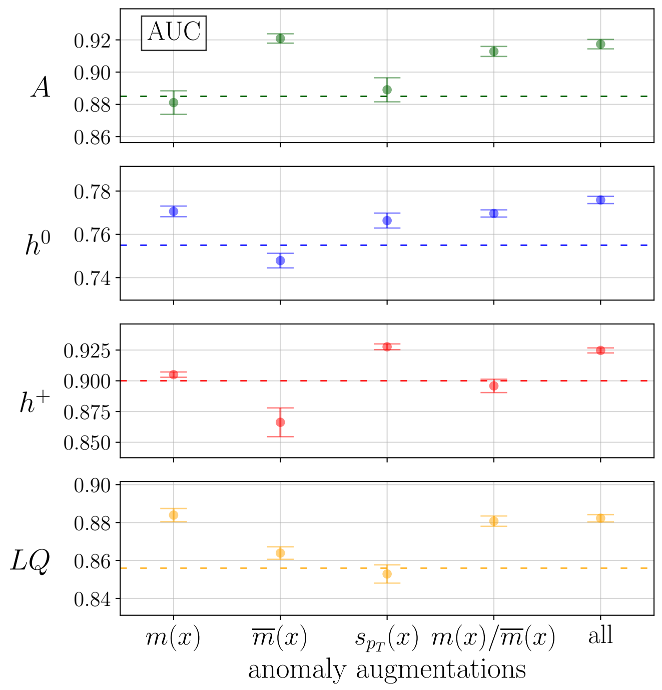


Figure 5.15: Results of a scan on the anomaly-augmentations used with the $\mathcal{L}^{+}_{\text{AnomCLR}}$ loss function. The augmentations are defined in Sec. 5.3.2. The dashed lines here correspond to the AutoEncoder on raw data baseline performance.

We can in fact see from this figure which augmentations are most advantageous for each signal. For the LQ this is the $m(x)$ augmentation, for h^0 it is both the $m(x)$ and s_{p_T} augmentations, for A it is the $\overline{m}(x)$ augmentation, and for h^+ it is the s_{p_T} augmentation. The important take away here is that in the case where we do not know what the signal is, including all augmentations will allow us to be signal-agnostic and retain the discriminative power for each signal type.

	Signal	AE-Raw	CLR	AnomCLR	AnomCLR+
AUC	A	0.885(2)	0.89(1)	**0.918(2)**	**0.917(3)**
	h^0	0.755(2)	0.726(9)	0.749(3)	**0.776(2)**
	h^+	0.900(4)	0.84(1)	0.898(3)	**0.925(2)**
	LQ	0.856(2)	0.82(1)	0.847(6)	**0.882(2)**
$\epsilon_b^{-1}(\epsilon_s = 0.3)$	A	47(2)	170(70)	**400(100)**	**270(50)**
	h^0	14.9(7)	10(1)	15.0(5)	**19.1(7)**
	h^+	60(10)	20(2)	53(3)	**110(10)**
	LQ	24.4(6)	16(1)	27(1)	**37(2)**
$\text{SI}(\epsilon_s = 0.3)$	A	2.05(5)	4.0(8)	**6.4(9)**	**5.0(4)**
	h^0	1.16(3)	1.01(5)	1.19(2)	**1.34(2)**
	h^+	2.3(2)	1.38(9)	2.24(7)	**3.2(2)**
	LQ	1.48(2)	1.25(5)	1.59(3)	**1.87(6)**

Table 5.2: Comparison of the different CLR loss functions, with and without anomaly-augmentations, and the AE trained on raw data.

The effect of representation dimension With CLR we can project our raw data from $\mathcal{D}$ to a representation of any dimension we like. We would expect that the larger the representation dimension the more information that can be encoded in the space. However we also expect that this would plateau or even peak at some point, and this what we want to investigate here. For this we use just $\mathcal{L}^{+}_{\text{AnomCLR}}$, however we expect the results with $\mathcal{L}_{\text{AnomCLR}}$ to be similar. Here we also obtain the error estimate from a combination of five CLR models and three AutoEncoders trained on each.

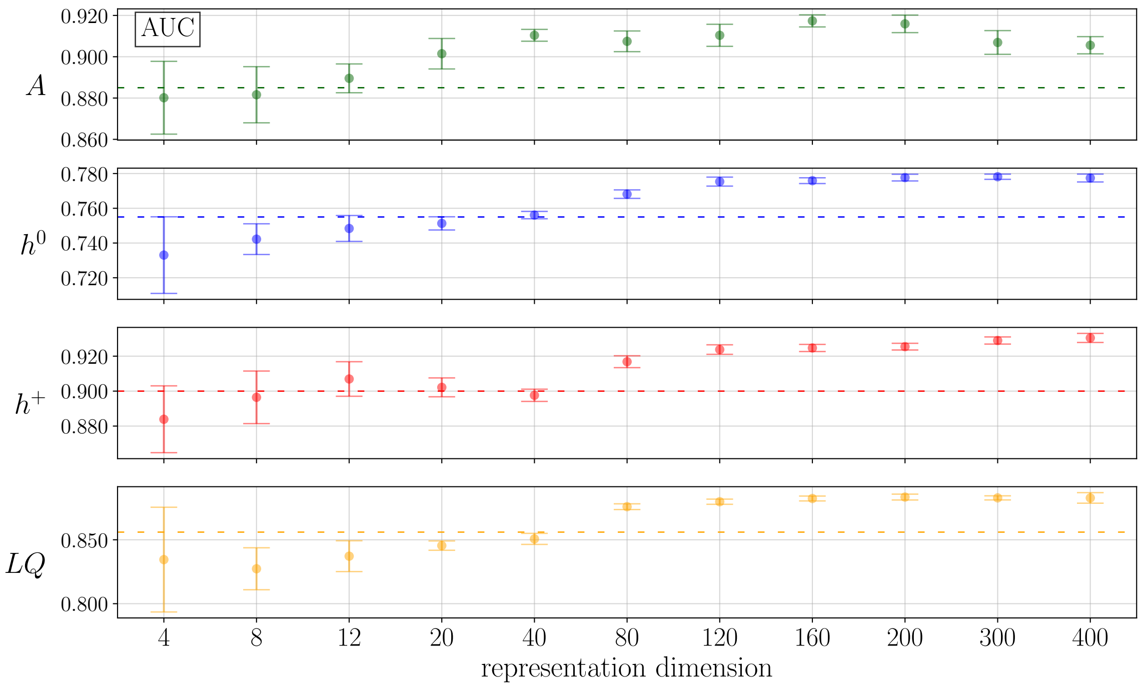


Figure 5.16: Results of a scan on the representation dimension used with the $\mathcal{L}^{+}_{\text{AnomCLR}}$ loss function. The dashed lines here correspond to the AutoEncoder on raw data baseline performance.

In Fig. 5.16 we see that increasing the representation dimension certainly improves the performance of the anomaly detection, at least up until a certain point. The A signal appears to achieve peak performance somewhere between dimensions 120 and 200, while the other signals performances all increase and/or plateau right up until 400. There is no fundamental limitation related to the representation size which we would expect to cause a degradation at larger dimensions, however there are two points we should keep in mind here. The first is simple, these means and variances are calculated with a total of fifteen runs (five CLR models each with three AutoEncoders), so more runs might present a clearer picture. The second point is that we have not optimised the AutoEncoder architecture or hyper-parameters as the representation size increases. While it is beyond the scope of this paper, it is possible that an independent hyper-parameter optimisation for each representation dimension would improve these results, particularly at larger dimensions. What these results show is that there is a clear tendancy for the results to improve as we increase from dimensions of ~ 4 to ~ 100, as we would naturally expect.

5.3.6 DarkCLR results

In this section, we show results using DarkCLR on the benchmark “Aachen” signal. First, we compare our results with previous methods tested on the same dataset. We then perform studies to test the robustness of our results with respect to variation of the semi-visible jet model parameters. Finally, we discuss the dependence of the performance on the main network parameters.

Improved performance First, we discuss the base pipeline of our procedure and compare the results with other methods. We train the transformer encoder network with the hyper-parameters as specified in Appendix B. The chosen embedding space uses 512 dimensions, and the augmentations follow the implementation described in Sec. 5.3.2, where $p_{\text{drop}} = 0.5$. Note that the size of the embedding space must be large enough to contain the information passed from the head to the output layer. As we show in Appendix A, our results are not sensitive to the specific choice of the embedding dimension, as long as it is sufficiently large. We show ROC curves for the CLR latent score s_{CLR} and the NAE score s_{NAE}. In addition, we report the low signal efficiency background rejection as a measure of the purity of a signal sample in the low background region and the AUC score. The error bands on s_{CLR} are taken from 5 runs of CLR training with different initializations. From each of these representations, we train 3 autoencoders for a total of 15 s_{NAE} scores, which are used to compute the mean and standard deviation. Note that no transformations are applied to the representations before training the autoencoder, thus limiting the preprocessing to the mere p_T rescaling and the physically guided CLR transformation.

Fig. 5.17 shows the ROC curves obtained with our method. The new embedding space greatly improves the background rejection ϵ_B^{-1}, in particular in the region of low signal efficiency as estimated by $\epsilon_B^{-1}(\epsilon_S = 0.2)$. We find that the transformer network does indeed encode information in the norm to discriminate between jets. In particular, it improves purity in the low background region, as shown by the background rejection of s_{CLR} at low signal efficiency. However, due to the high dimensionality of the representations, many jets will share the same norm in the bulk of the distribution, causing the s_{CLR} ROC curve to drop off at $\epsilon_S = 0.3$. We also observed similar problems when training a standard autoencoder. This is solved by a more precise density estimator like the NAE. The resulting s_{NAE} ROC curve is much more stable with an average AUC of 0.76 and a $\epsilon_B^{-1}(\epsilon_S = 0.2) = 59$.

Tab. 5.3 summarizes the AUC and the background rejection $\epsilon_B^{-1}(\epsilon_S = 0.2)$ for DarkCLR and compares them to previous methods: an NAE trained on jet images [1], a Dirichlet variational autoencoder [180], and an invertible neural network [58]. While the best AUC is similar for all methods, with DarkCLR we find much stronger background rejection at low signal efficiency, and we do not rely on image-based representations or any specific preprocessing steps.

Dependence on the dark shower signal As a next step, we study the robustness of our method with respect to the main phenomenological parameters of the semi-visible jet as described in Sec. 4.1. We set up a benchmark by training a transformer classifier with 100k jets equally divided between the QCD background and the “Aachen” dataset. We then use the classifier score to detect the signals with different invisible fraction r_{inv} and dark meson mass scale m_{mesons}. The classifier uses the same backbone transformer

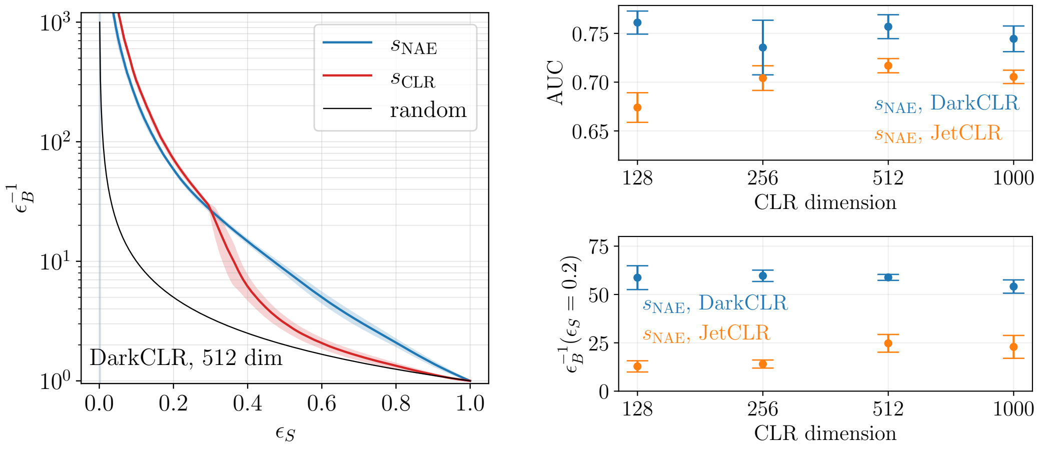


Figure 5.17: (left) ROC curves of background suppression ϵ_B^{-1} versus signal efficiency ϵ_S, computed from the L_2 norm of the representations, s_{CLR} (red), and from the MSE of the NAE trained on the representations from DarkCLR, s_{NAE} (blue). (right) CLR and NAE AUC (upper panel) and background rejection at low signal efficiency (lower panel) for different embedding dimensions.

	DVAE [180]	INN [58]	NAE Jet images [1]	DarkCLR
AUC	0.71	0.73	0.76(1)	0.76(1)
$\epsilon_B^{-1}(\epsilon_S = 0.2)$	36	39	41(1)	59(1)

Table 5.3: Summary of AUCs and background rejections at low signal efficiencies for DarkCLR compared to other methods. The numbers in parenthesis indicate the standard deviation of the score from an ensemble of networks. For the DVAE and the INN this was not reported.

architecture of Sec. 5.3.3 where the head network is replaced by a two-layer MLP with ReLU nonlinearities and a single output. We train the network for 300 epochs, minimizing the binay cross-entropy loss, and refer to the validation loss to select the best model.

Fig. 5.18 shows the results of the supervised classifier (left panel) compared to DarkCLR trained only on the QCD background and tested on all signals (right panel). The supervised classifier shows a large drop in performance when applied to datasets with different model parameters, see also [184]. Instead, our DarkCLR method performs well on different semi-visible jet signals, as expected from the unsupervised training approach.

The small differences between the DarkCLR ROC curves for the various signals can be understood by analyzing the phenomenological aspects of the different semi-visible jet models. As we reduce the invisible fraction r_{inv}, the signal becomes more similar to a QCD jet, increasing the overlap between the two distributions and thus reducing the detection efficiency. Similarly, increasing the confinement scale and thus the mass of the dark hadrons leads to an earlier hadronization of the dark quarks. Therefore, the visible SM decays continue to shower down to the QCD confinement scale, again more closely resembling a QCD background jet initiated by light quarks. We observe this effect when we increase the energy scale from the default choice of the Aachen benchmark dataset to $m_{\pi_d} = m_{\rho_d} = \Lambda = 10\,\mathrm{GeV}$ and $20\,\mathrm{GeV}$.

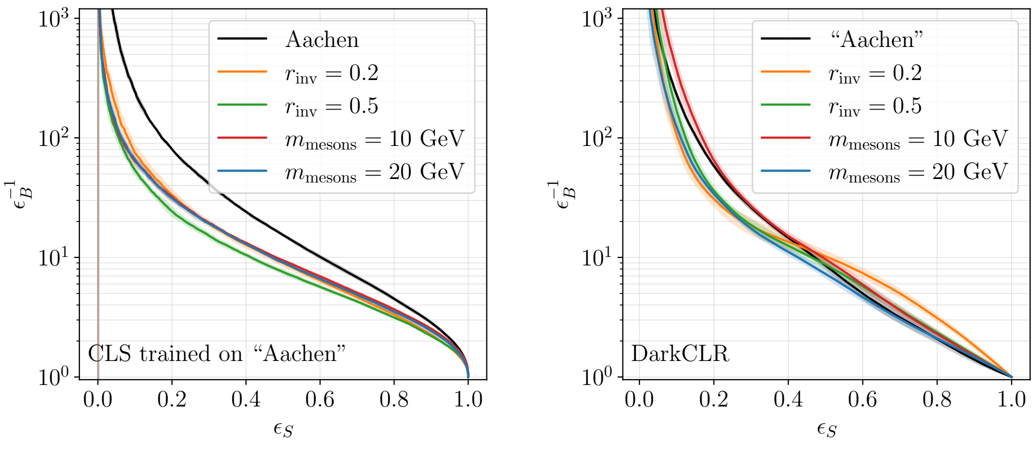


Figure 5.18: Left panel: ROC curves of a supervised classifier trained on the “Aachen” benchmark signal and tested on datasets with different dark shower model parameters. Right panel: ROC curves obtained from DarkCLR after training on the QCD background only and tested on additional datasets.

	$\epsilon_B^{-1}(\epsilon_S = 0.2)$				
	”Aachen”	$r_{\mathrm{inv}} = 0.2$	$r_{\mathrm{inv}} = 0.5$	$m_{\mathrm{mesons}} = 10$ GeV	$m_{\mathrm{mesons}} = 20$ GeV
CLS (“Aachen”)	80(1)	28(2)	22(2)	30(2)	28(2)
DarkCLR	58(2)	28(2)	35(3)	65(7)	33(1)

Table 5.4: Summary of the results presented in Fig. 5.18 for the background rejection ϵ_B^{-1} at a signal efficiency of $\epsilon_S = 0.1$.

For a summary of the background suppression at low signal efficiency, see Tab. 5.4. The generalization capabilities of DarkCLR outperform the supervised classifier for all signal models, especially in the more interesting low signal efficiency region.

Impact of Anomaly augmentation To validate the use of anomalous augmentations, we compare DarkCLR with the standard JetCLR training. The latter is trained only on QCD jets using the set of physical augmentations. We refer to previous work for the implementation and training of JetCLR [198]. After creating the new representations, we train an NAE using the same procedure. Fig. 5.17 shows the performance of JetCLR compared to DarkCLR in terms of AUC and background rejection for the benchmark dataset. Without anomalous pairs, the results vary between different embeddings and underperform in both figures of merit. Notably, DarkCLR improves detection at low signal efficiency even for small embedding dimensions, while without augmentation we observe a small increase in sensitivity only for large embedding spaces.

CHAPTER **6**

Summary and outlook

The LHC is entering a new data-taking phase which will provide an enormous amount of data to analyse. The success of this final phase will be dictated by our ability to understand the dataset to the best of our capabilities. The last decade of precise measurements at the LHC is already paving the way for future analysis. A key ingredient to further extend this success is the development of new techniques which can boost LHC analysis. Novel machine learning techniques fit the bill in every aspect. The possibility of modelling high-dimensional spaces with a reduced number of assumptions is an appealing face of ML which is already catching on at the LHC.

In this thesis, we explored ML solutions to real problems we face and will be facing at the LHC. Starting from the importance of precise, accurate, and fast simulations, we propose fast surrogate networks to replace the most computationally expensive step of the simulation chain, calorimeter simulations. Indeed, calorimeter showers are one of the most exciting applications of modern generative networks in fundamental physics. Their specific challenge is the high dimensionality of the voxelized phase space, combined with extremely sparse data and an LHC-level precision requirement. In our case, we focused on datasets of increasing target phase space dimensionality, from $\mathcal{O}(100)$ to $\mathcal{O}(10000)$ input features.

We started with a normalizing flow architecture and a VAE to reduce the dimensionality with a learned latent space. For the simplest case, the dataset 1 photons, we have found that the INN generated high-fidelity showers and learns the phase space density of high-level features at the 10% level, except for failure modes which we can identify using high-level features and classifier weights over the low-level phase space. For this dataset, the VAE+INN shows no advantage, but less expressivity for example affecting the sparsity. For the pions in dataset 1, the INN faces more serious challenges, including mis-modelled features, and a wider range of learned classifier weights. The performance of the INN and the VAE+INN becomes much more similar. These modelling limitations are counterbalanced by the generation speed of the network. The fully invertible architecture allows us to generate showers in $\mathcal{O}(1)$ ms, which is a huge speed-up factor compared to Geant4 simulations.

Exploring the other side of the trade-off between speed and accuracy. Diffusion networks allow us to go a step beyond standard normalizing flows. Our CaloDREAM architecture first factorizes the generation of detector showers into an energy network and a shape network. Both networks are trained using Conditional Flow Matching. The former generates the layer energies using a transformer backbone with self-attention and cross-attention blocks. For the latter, we use a 3-dimensional vision transformer, operating on patches of the target phase space. Because diffusion networks are slower than alternative generative networks, we use bespoke samplers to enhance their generation speed, at no cost of the precision and improving the fidelity in case of limited resources. We can use a VAE to reduce the dimensionality using latent diffusion. We find essentially no loss in performance, except for the reproduction of low-energy voxels and, with it, sparsity, which can be improved by introducing a MeV-level energy threshold. However, further studies are needed to understand the effects of mapping the distributions into real detectors with irregular geometries, more complex distributions from different incident particles, e.g. hadrons, and varying angles of impact.

Our study shows that modern generative networks can be used to describe calorimeter showers in highly granular calorimeters. When the number of phase space dimensions becomes very large and the data becomes sparse, a latent diffusion network combined with an (autoregressive) transformer and bespoke sampling provides excellent benchmarks in speed and precision.

The second part of the thesis focuses on BSM searches. In particular, we explore new ML-based techniques for model-agnostic searches. We define anomalous objects based on their likelihood, expecting BSM physics to populate a tail of the SM background distribution. We first present the NAE for high-energy physics. The NAE combines a standard autoencoder architecture with an energy-based normalization in the loss. This means it constructs an energy or MSE landscape such that any anomaly with features not present in the training data will be pushed to even larger energies. Because of the normalization, the NAE can also balance different kinds of features, which means that the absence of a background feature in signal jets will be visible in the energy landscape. The additional components of the NAE architecture do not increase the size of the network or the inference time, they only increase the time taken to train the model. Technically, we adjust the training after an AE pre-training step. Applied to top vs QCD jets we first show that for this extreme case of different compressibilities the NAE still tags complex top jets in a simple QCD background as well as simple QCD jets in a complex QCD background. For the more challenging Aachen and Heidelberg dark jets the NAE works for a reasonable single choice of preprocessing. The performance gain from using different preprocessing on the two datasets are explainable in terms of the changes induced by the features of the two signals.

Furthermore, we defined a procedure to construct observables without using hand-crafted preprocessing steps but rather imposing approximate invariance under transformations of the data. In this contrastive learning scheme, we use anomalous augmentations of collider data to build a representation space from which to construct anomaly scores with a range of methods, for example, using autoencoders, that are approximately invariant under the transformations. This is a self-supervised method, based on the contrastive learning idea. We tested this method on the CMS ADC dataset and semi-visible jets, and compared to the raw data baselines we found large improvements on all signals.

In AnomalyCLR and DarkCLR, the anomaly-augmented data is constructed from the background data through feature augmentation, designed to emulate a generic anomaly.

We have discussed in detail how we do this for the reconstructed event-level anomalies and in studies of jet substructures. We proposed a new loss function which we use to train a deep transformer-based neural network. This network projects the events to a new representation, in which the anomaly-augmented events are far from their original counterparts while being close to similar events. The network then learns a highly discriminative representation of the events which is sensitive to the presence of potential anomalies. We have seen that the choice of these augmentations is quite model-agnostic. This model-agnostic nature of the approach can be seen in how the results improve across all four signals considered.

In addition in DarkCLR, we test the dependence of our network on the main phenomenological parameters entering the dark shower model, the invisible fraction of particles and the mass of dark mesons. We find that a supervised classifier is highly sensitive to the specific choice of signal parameters used during training, especially at low signal efficiencies. In contrast, our method, based on a density estimation of the background, is more robust to a variation of the parameters of the dark shower model, thus validating the application of unsupervised methods for a model-agnostic search. In our experiments, we assumed uncorrelated visible constituents, i.e. constituents are uniformly dropped in the jet. A dedicated study on this specific signature is needed to evaluate any potential bias. However, our framework is flexible enough to account for these modifications. Both positive and anomalous augmentations can be extended to cover different transformations.

We conclude by remarking on the power of ML tools for HEP. While the underlying theme in the thesis has been density estimation and generative networks, we discussed the strong impact that other tools, i.e. classification and representation learning, can have at the LHC. Likely, ML will become the standard at the LHC with a cohesive interplay of different methods without a clear winner from the standoff.

APPENDIX **A**

Supplementary material

Bayesian CaloINN on CaloGAN dataset

In this section we discuss the INN performance on the even simpler CaloGAN dataset [12, 14]. The INN architecture is described in Sec. 4.3. To extract uncertainties from the generative network, we promote the deterministic INN to its Bayesian counterpart [43,168]. The implementation follows the variational approximation substituting the linear layer with a mixture of uncorrelated Gaussians with learnable means and a diagonal covariance matrix. In practice, we only upgrade the last layer of each sub-network to a Bayesian layer [201].

Figure A.1 showcases two high–level features as examples of the performance of the CaloINN as compared to the training data distribution generated by Geant4. We show the brightest voxel distribution in layer 0, the average ϕ location of the showers in layer 2, and the width of the shower depth width defined as the standard deviation of s_d [21], with

$$s_d = \frac{\sum_{k=0}^{2} kE_k}{\sum_{k=0}^{2} E_k}. \tag{A.1}$$

The error bars in the Geant4 distribution are the statistical errors while for the INN we estimate the uncertainties by sampling $N = 50$ times from the network and resampling the network parameters each time.

To evaluate our model on low-level observables, we resort again to classifier-based metrics. As already studied in a previous work [2], the INN samples are indistinguishable from the Geant4 counterpart besides a few specific phase-space regions. We train a classifier on the CaloFlow samples and find a large tail towards small weights. From clustering of the tail, we observe a clear dependence on the energy deposition total energy deposition. We link this effect to the learned energy variable $u_2 = E_1/(E_1 + E_2)$ and the noise injection procedure. If the noise is added at voxel-level, before calculating the additional energy variables, the flow learns distorted energy ratio distributions. Especially in the last layer, where the average energy deposition is smaller, this effect is larger. We

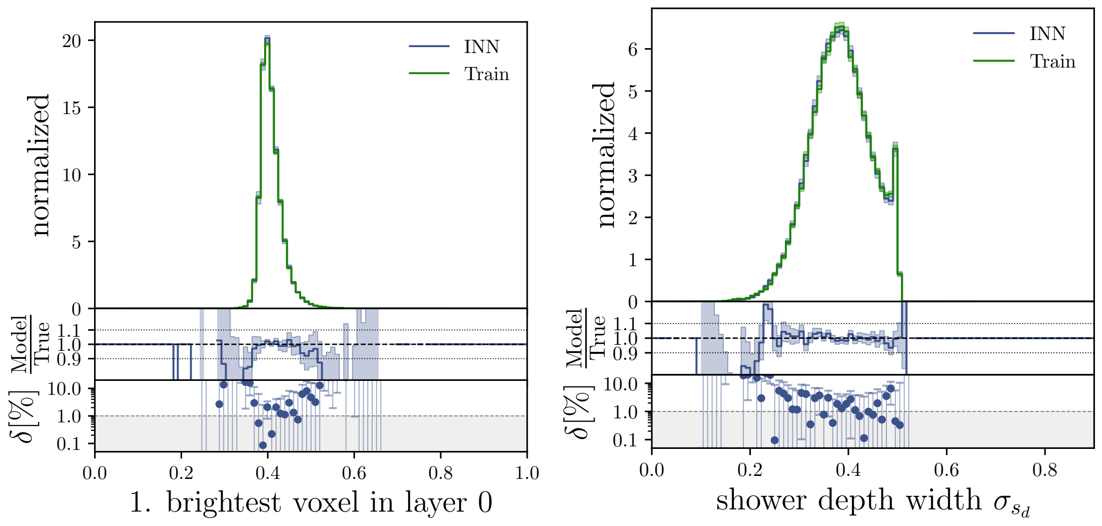


Figure A.1: Comparison between CaloINN and GEANT4 on three high level features. Brightest voxel distribution in layer-0 (left), ϕ coordinate of the center of the shower in layer-2 (right), and width of the shower depth (bottom). Error bars on the INN are calculated after sampling from the Bayesian network $N = 50$ times.

summarize this effect in Fig. A.2. We also provide the AUCs and the generation timings in Tab. A.1.

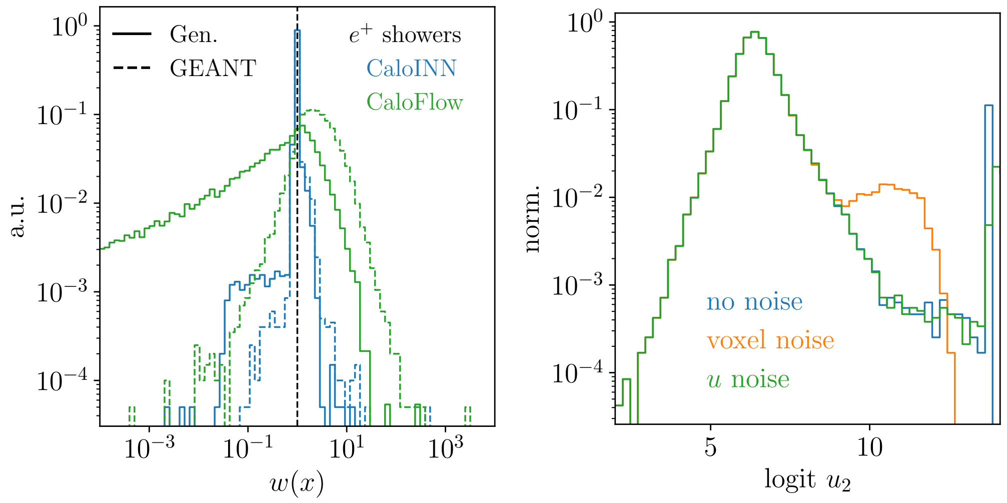


Figure A.2: (left) Weight distribution of CaloFlow and CaloINN for e^+ showers. (right) u_2 distribution with different noise injections.

AUC		CaloFlow [21]	CaloINN
e^+	unnorm.	0.859(10)	0.525(2)
	norm.	0.870(2)	0.598(3)
	hlf	0.795(1)	0.656(2)
γ	unnorm.	0.756(50)	0.530(2)
	norm.	0.796(2)	0.584(2)
	hlf	0.727(2)	0.671(2)
π^+	unnorm.	0.649(3)	0.662(2)
	norm.	0.755(3)	0.735(4)
	hlf	0.888(1)	0.786(4)

	Batch size	CaloFlow [23]	CaloINN
GPU	1	$55.12 \pm 0.19^*$	$23.79 \pm 0.10^*$
	100	0.744 ± 0.04	0.425 ± 0.005
	10000	0.249 ± 0.003	0.211 ± 0.003
CPU	1	$119.9 \pm 0.9^*$	$46.39 \pm 3.18^*$
	100	3.13 ± 0.11	1.14 ± 0.03
	10000	1.681 ± 0.004	0.72 ± 0.01

Table A.1: (left) AUC of the two classifiers trained on the CaloFlow teacher and CaloINN samples. (right) Per shower generation timings in ms. We show mean and standard deviation of 10 independent runs of generating 100k showers. The star indicates that only 10k samples were generated in total.

Manifold learning with Bernoulli VAE

The VAE introduced in Sec. 4.4.1 is trained separately, using the BCE reconstruction loss

$$\mathcal{L}_{\text{VAE}} = -\big\langle x \log(x_\psi) + (1-x)\log(1-x_\psi)\big\rangle_{p_\psi(r|x)} + \beta D_{\text{KL}}[p_\psi(r|x), \mathcal{N}(0,1)] \; . \tag{A.2}$$

This loss provides notably better reconstruction quality than the standard MSE loss, both in terms of high-level features and a neural network classifier trained to distinguish reconstructed showers from an independent test set. A detailed description of the network architecture is provided in Tab. B.6. Each block consists of three Conv2d operations that preserve the number of channels of which the final one downsamples according to the stride and padding parameters. In addition, we break the translation equivariance by adding the coordinates of each input to the activation map as new channels [38].

In Fig. A.3 we provide a set of kinematic distributions similar to Fig. 4.14 for DS3, to illustrate the VAE reconstruction. We find that the only missing feature in the learned manifold is the distribution of the low-energetic voxels, also reflected in the sparsity. We also train a classifier using the hyperparameters of Tab. B.5 on the low-level features which gives an AUC score of 0.512(5) consistently for both, DS2 and DS3.

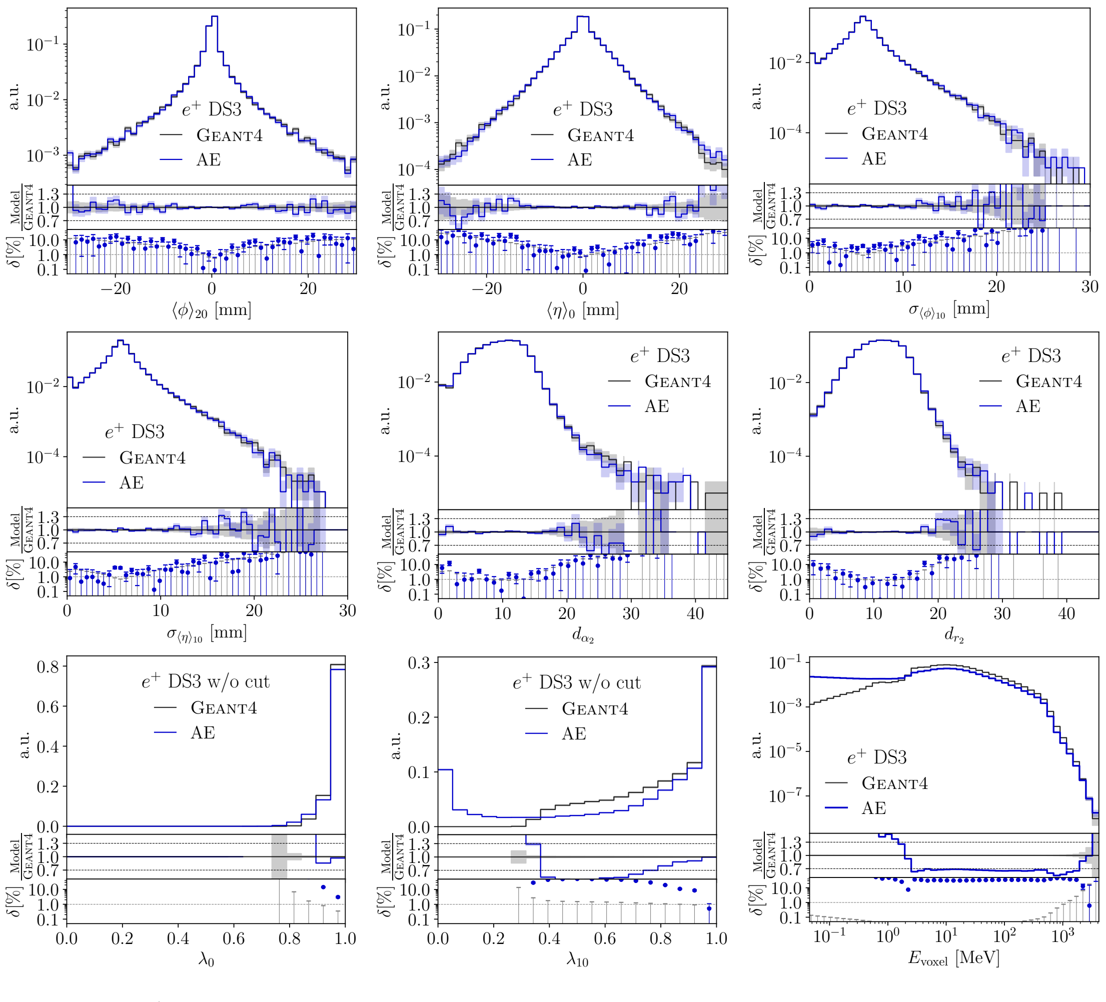


Figure A.3: Selection of high-level features sensitive to the reconstruction of the autoencoder for DS3.

Shower observables with weight clustering

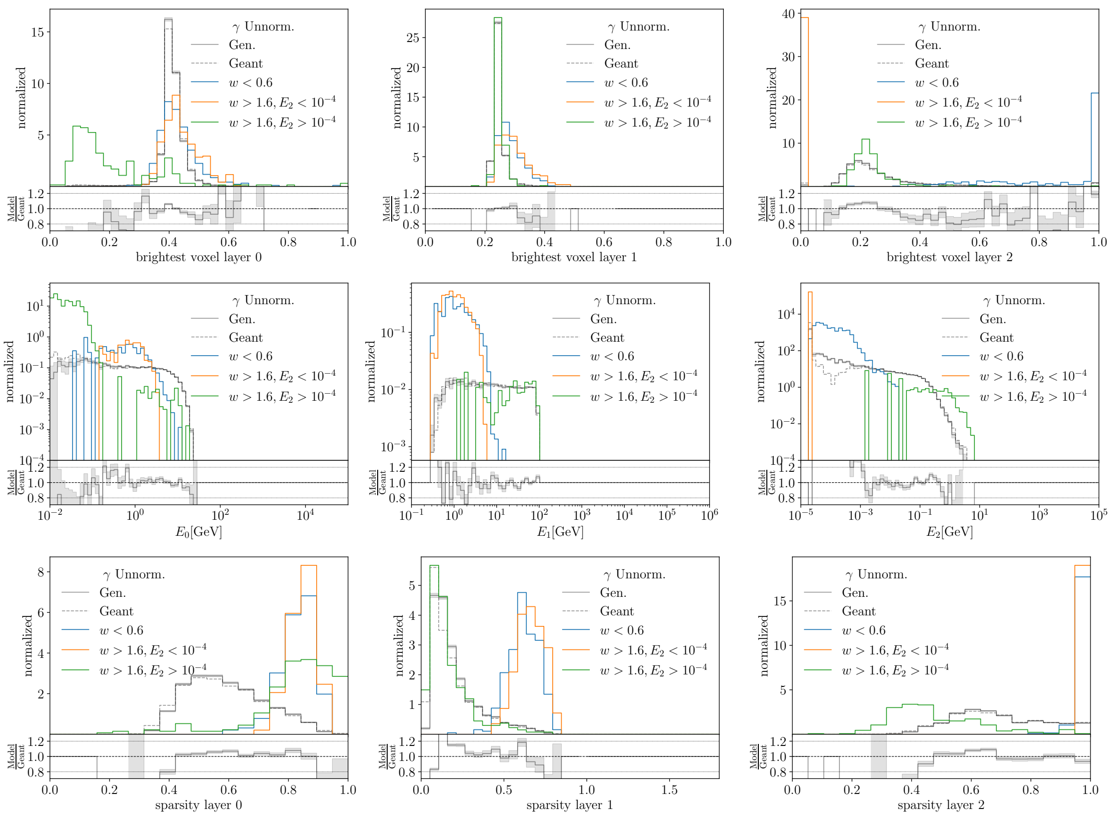


Figure A.4: Clustering plots for γ: (i) reweighting is required at low energies, but the pattern is not just the energy; (ii) (orange) under-sampling of soft showers with zero energy deposition in layer-2; (iii) (blue) induced over-sampling of soft showers in layer-2; (iv) (green) under-sampling of delayed showers, low energy deposition in layer-0.

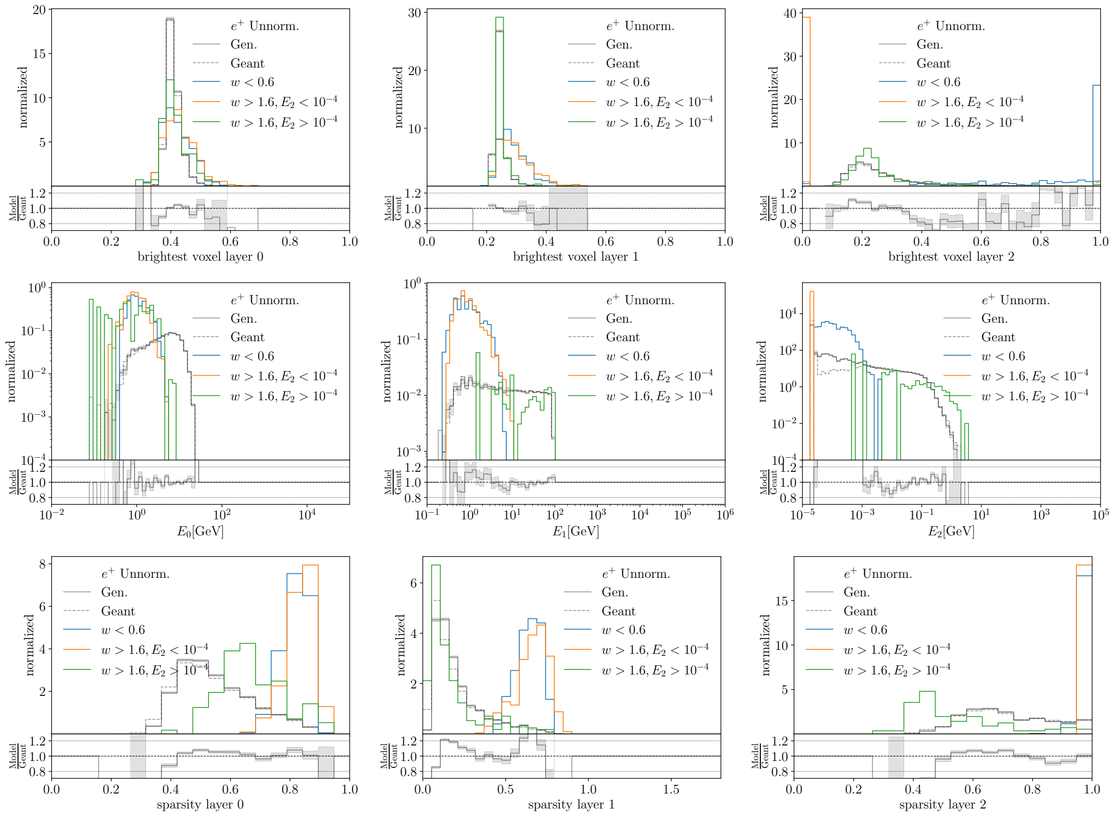


Figure A.5: Clustering plots for e^+: similar pattern of γ showers, expected given the similar physics and data structure.

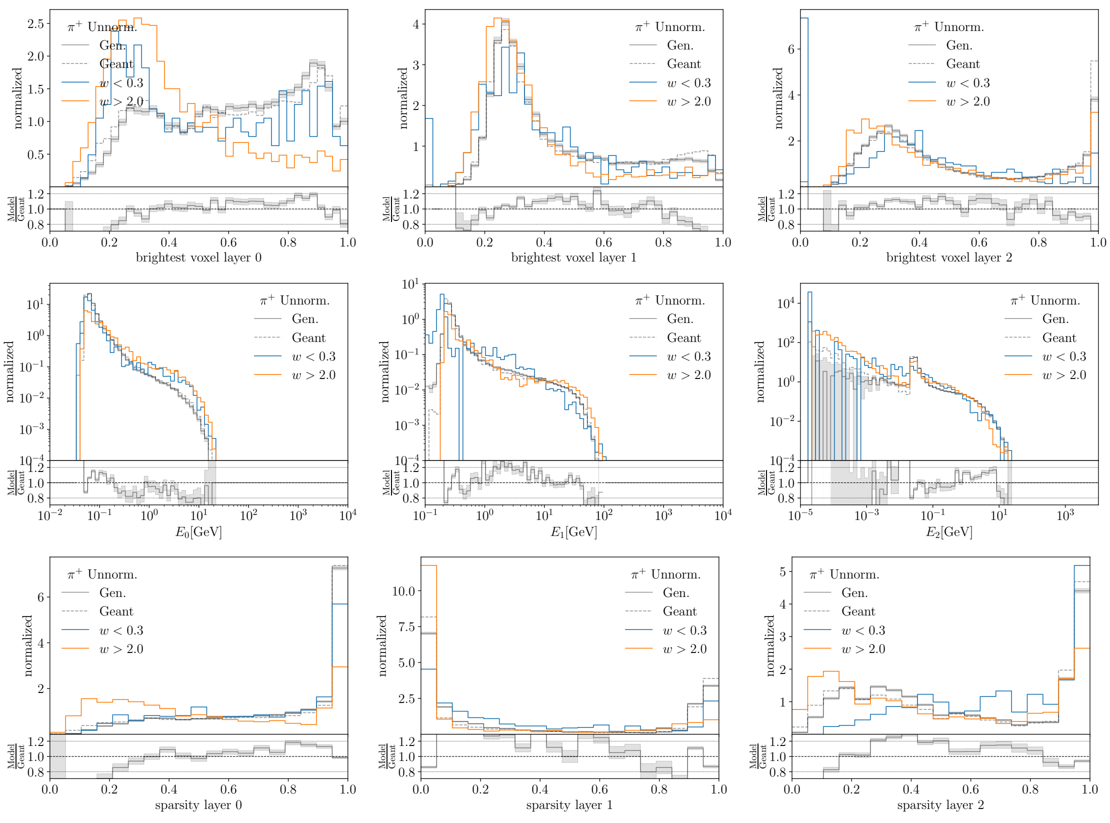


Figure A.6: Clustering plots for π^+: (i) for the different energies the INN finds all features, but the balance between feature and continuum is not perfect; (ii) in both tails corrections at all energies are applied; (iii) the generator over-samples showers with no energy deposition in layer-1 and layer-2; (iv) large sparsity values are underestimated by the INN.

Linear classifier test of CLR representations

We study the separability between the QCD and semi-visible jets representations in a supervised way by training a linear classifier test (LCT) between background and signal. Even though we move to a supervised scenario, the network never accesses the signal data during training. This evaluation will test the separation power and the information content in the representations starting only from QCD jets and their augmentations. We disentangle the effects of the embedding dimension and the head network by selecting 128 as the embedding dimension of the transformer and scanning over the output dimension of the head network. This choice closely matches the original dimensionality of the input data. Fig. A.7 (left) shows that the LCT of the head representation is informative regardless of the output dimension. The head network is affected by the projection on the hypersphere and requires a larger dimension to saturate to the same separation power. In both cases, we observe that the representation space is simpler than the original constituent-level space. The implemented LCT is a single linear layer network without non-linearities.

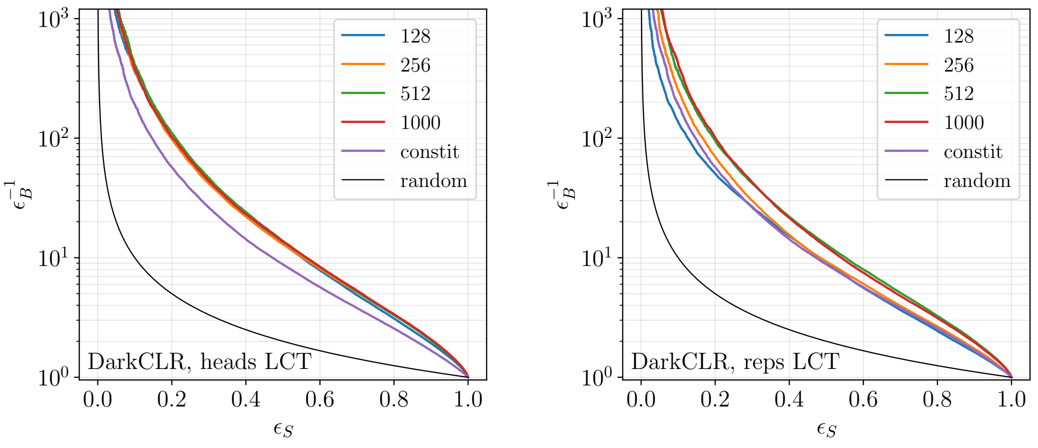


Figure A.7: Linear Classifier test between the Aachen benchmark dataset and QCD jets. Head representations (left) and output representations (right) with different embedding dimensions from 128 up to 1000. The LCT on raw constituents is shown in purple.

APPENDIX B

Hyperparameters

CaloINN

In this appendix we give some details on the network architectures and the preprocessing. The INN and the VAE+INN take layers normalized by the layer energy as input. The extra energy dimensions, calculated as in Eq. 4.2, are appended to the feature vector.

In the INN, we apply uniform noise and and a regularized logarithmic transformation with strength α. The transformation applied to the features is a rational quadratic spline [148] for dataset 1 and a cubic spline [149] for dataset 2. The prediction of the spline parameters is obtained with an feed-forward sub-network with 256 nodes for each hidden layer. To equally learn each dimension, we permute the order of the features after a transformation and normalize the output to mean zero and unit standard deviation with an ActNorm [49] layer. In the large-scale architecture, we stack twelve blocks to construct the INN with the additional preprocessing block.

The VAE preprocessing has a similar structure. After normalization, we apply an α-regularized logit transformation and a normalization to zero mean and unit standard deviation to each feature. We do not add noise during training and we set the latent dimension to 50 for dataset 1 and 2, and to 300 for dataset 3. We provide the full list of parameters in Tabs. B.1 and B.2.

The classifiers, trained for the evaluation of the generative networks, are simple MLP networks with leaky ReLU. We use three layers with 512 nodes each and a batch size of 1000. The network is trained for 200 epochs with a learning rate of $2 \cdot 10^{-4}$ and the Adam optimizer with standard parameters. To prevent overfitting, especially for the larger datasets, we apply 30% dropout to each layer, and we reduce the learning rate on plateau with a decay factor of 0.1 and decay patience of 10. The splitting between training, validation, and testing is 60/20/20%. The selection of the best network is based on the best validation loss.

Parameter	INN DS1/DS2	INN (with VAE)
coupling blocks	RQS / Cubic	RQS
# layers	4 / 3	3
hidden dimension	256	32
# of bins	10	10
# of blocks	12/14	18
# of epochs	450 / 200	200
batch size	512 / 256	256
lr scheduler	one cycle	one cycle
max. lr	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
$\beta_{1,2}$ (ADAM)	$(0.9, 0.999)$	$(0.9, 0.999)$
b	$5 \cdot 10^{-6}$	/
α	$1 \cdot 10^{-8}$	$1 \cdot 10^{-6}$

Table B.1: Network and training parameters for the pure INN.

Parameter	VAE	
lr scheduler	Constant LR	Inner VAE
lr	$1 \cdot 10^{-4}$	
hidden dimension	5000, 1000, 500 (Set 1)	
	1500, 1000, 500 (Set 2)	
	2000, 1000, 500 (Set 3)	
latent dimension	50 (Set 1,2) / 300 (Set 3)	
# of epochs	1000	
batch size	256	
β	$1 \cdot 10^{-9}$	
threshold t [keV]	2 (Set 1) / 15.15 (Set 2,3)	
hidden dimension	1500, 800, 300	Kernel
kernel size	7	
kernel stride	3 (Set 2), 5 (Set 3)	

Table B.2: Network and training parameters for the VAE-INN.

CaloDREAM

Parameter	DS2 & DS3
Epochs	500
LR sched.	cosine
Max LR	10^{-3}
Batch size	4096
ODE solver	Runge-Kutta 4 (50 steps)
Network	transformer
Dim embedding	64
Intermediate dim	1024
Num heads	4
Num layers	4
Network	dense feed-forward
Intermediate dim	256
Num layers	8
Activation	SiLU

Table B.3: Parameters for the autoregressive energy network in Sec. 4.4.1.

	ViT		laViT	
Parameter	DS2	DS3	DS2	DS3
Patch size	(3, 16, 1)	(3, 5, 2)	(3, 1, 1)	(3, 2, 2)
Embedding dimension	480	240	240	240
Attention heads	6	6	6	6
MLP hidden dimension	1920	720	960	960
Blocks	6	6	10	10
epochs	800	600	800	400
batch size	64	64	128	128
LR sched.	cosine			
Max LR	10^{-3}			
ODE solver	Runge-Kutta 4 (20 steps)			

Table B.4: Parameters for the shape networks in Sec. 4.4.1, for the full and the latent space.

Parameter	Value
Optimizer	Adam
Learning rate	$2 \cdot 10^{-4}$
LR schedule	reduce on plateau
Decay factor	0.05
Decay patience (epochs)	20
Batch size	1000
Epochs	200
Number of layers	3
Hidden nodes	512
Dropout	20%
Activation function	leaky ReLU
Training samples	70k
Validation samples	10k
Testing samples	20k

Table B.5: Parameters for the classifier network used to calculate the weights of Fig. 4.17.

Parameter	Value	
	DS2	DS3
Loss	BCE + βKL	
β	10^{-6}	
Epochs	200	
Out activation	sigmoid	
Lr sched.	OneCycle	
Max lr	10^{-3}	
# of blocks	2 (+ bottleneck)	
Channels	(64, 64, 2)	
Dim. bottleneck	(2, 15, 9, 9)	(2, 9, 26, 16)
Kernels	[(3,2,1), (1,1,1)]	[(5,2,3), (1,1,1)]
Strides	[(3,2,1), (1,1,1)]	[(2,2,1), (1,1,1)]
Paddings	[(0,1,0), (0,0,0)]	[(0,1,0), (0,0,0)]
Normalized cut	$1 \cdot 10^{-6}$	

Table B.6: Parameters of the autoencoder for DS2 and DS3 used for the laViT network in Sec. 4.4.1.

Parameter	Value
Reference solver	midpoint (100 steps)
Initialization	Euler
Optimizer	Adam
Learning rate	$1 \cdot 10^{-3}$
Batch size	100
Max iterations	5000
Stopping patience (iterations)	200

Table B.7: Parameters used to train BNS solvers, described in Sec. 4.4.1.

Parameter	Value
Optimizer	Adam
Learning rate	0.001
LR schedule	reduce on plateau
Decay factor	0.1
Decay patience (epochs)	5
Batch size	1000
Epochs	150
Number of layers	3
Hidden nodes	512
Dropout	10%
Activation function	leaky ReLU
Training samples	60k
Validation samples	20k
Testing samples	20k

Table B.8: Hyperparameters of the classifier network used to generate the clustering plots in Sec. 4.5.

NAE

In this section we provide the details of the network architecture and the parameters setting for the NAE trained in Sec. 5.2. Tab. B.9 summarizes the layers used in both encoder and decoder and their parameters. The architecture of the AE is summarized in Tab. B.9. Each layer is regularized using Spectral Normalization. The output of the last encoder layer is mapped to the surface of a hyper-sphere $\mathbb{S}^{D_{\mathbf{z}}-1}$. During training, each step of the preliminary LMC is projected on the surface. The initial latent distribution is uniform but a buffer of size 10000 is used to store the final points of each chain. Then, the initial points are sampled from the uniform distribution with probability 0.05 or from the buffer with probability 0.95.

Additional regularization terms are used to improve training stability. The L2 norm of the weights for both encoder and decoder is added to the loss function with a coefficient 10^{-8}. We also prevent the negative energy divergence by adding the average squared energy of the training batch to the final loss function.

The bottleneck of the training procedure is the sampling algorithm. The parameters of the LMCs have been tuned to give fine samples after a small amount of steps to train a model in less than 15 hours. We have found that the structure of the initial manifold after pre-training plays an important role for the following NAE updates. A large batch size gave the best results while a smaller one during NAE showed more stable results.

Encoder	Conv2d(1, 8, 3, 1, 1, True) - PReLU - Conv2d(8, 8, 3, 1, 1, True) - PReLU - MaxPool2d(2, 2) - Conv2d(8, 8, 3, 1, 1, True) - PReLU - Conv2d(8, 8, 3, 1, 1, True) - PReLU - Conv2d(8, 1, 3, 1, 1, True) - PReLU - Flatten - Dense(400, 100, True) - PReLU - Dense(100, $D_{\mathbf{z}}$, True)
Decoder	Dense($D_{\mathbf{z}}$, 100, True) - PReLU - Dense(100, 400, True) - PReLU - Reshape(1, 20, 20) - Deconv2d(1, 8, 3, 1, 1, True) - PReLU - Deconv2d(8, 8, 3, 1, 1, True) - PReLU - Upsampling(2, 'b') - Deconv2d(8, 8, 3, 1, 1, True) - PReLU - Deconv2d(8, 1, 3, 1, 1, True) - Sigmoid

Table B.9: Architecture of the normalized autoencoder. The latent space is $D_{\mathbf{z}} = 3$.

Contrastive learning

hyper-parameter	
model (embedding) dimension	160
feed-forward hidden dimension	160
output dimension	160
# self-attention heads	4
# transformer layers (N)	4
# layers	2
dropout rate	0.1

hyper-parameter	
optimiser	Adam($\beta_1 = 0.9$, $\beta_2 = 0.999$)
learning rate	5×10^{-5}
batch size	128
# epochs	500

Table B.10: Default setup of the transformer-encoder network and the Anomaly-CLR training.

Hyper-parameter	Value
Embedding dimension (d_r)	128
Feed-forward hidden dimension (d_z)	512
Output dimension (d_z)	512
# self-attention heads	4
# transformer layers (N)	4
# head architecture layers	2
Dropout rate	0.1
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Learning rate	5×10^{-5}
Batch size	256
# constituents (N_c)	50
# jets	100k
# epochs	150

Table B.11: Default setup of the transformer-encoder network and the DarkCLR training.

Acknowledgements

I am profoundly grateful to Tilman Plehn for giving me the Key opportunity to be successful in life and, more importantly, for all the “Ciao!” shouted in a weird Italian accent. These three years would also Not be possible without the Opportunity Michael Krämer gave to me. Thank you for Welcoming me to Aachen and for teaching me the Need to look for clever Hanses in life.

I am also grateful to the referees of this thesis, Prof. Tilman Plehn and Prof. Fred Hamprecht, and the rest of the committee, Prof. Ulrich Uwer and Prof. Björn Malte Schäfer, for every Ounce of their Time for a defence on a Thursday, despite *all* the unwritten rules.

Jonas, Lorenz, Nathan, and Sofia, Thank you for making this thesis readable, and thanks to Nikita for providing the best German abstract I could ever Have imagined.

A special Thank you to everyone I have met during these years inside and outside the office. You helped me climb the Mountain called PhD. Although it was just three Years, all of you taught me something which is now a piece of myself.

Barry, thank you for welcoming me here in Heidelberg. Your presence was Essential at the beginning of my PhD. By the way, that Ireland vs Italy rugby match was close;

Michel, your Carbonara is one of the best. Obviously, the sauce Must have cream in It.

Emma, even if it is not easy to Navigate in academia, you taught us to be happy and kind researchers. I wish you all the best with your Great workshops.

Benedikt and Theo, ours is the Uttermost pasta competition in Europe.

Lennart, Thank you for bringing Markov chains into an office full of machine learning.

Theo and Nina, I hope you enjoyed our days on the Beach. Zia Bianca always wants to know how you two are doing;

Nikita, your laughs at people’s questions are very Enjoyable;

Lorenz, the “Blondie” with unmatched LaTeX skills. I hope you’ll get paid for all The work you do;

Claudius, thank you for the many tips during your stay in Heidelberg and the awesome GlühWien workshop;

Jonas, “Guten Morgen Sonnenschein” should be the world’s greatest Hit;

Ayo, we are the Aspen seniors. Snowmass, the discussion during the coffee breaks, the volleyball, and the jelly beans made that Trip unforgettable;

Nathan, you always surprised me with your Impactful questions straight to the point.

Javi, Thank you for explaining to us why the seafood in North Spain is better;

Sofia, you *will* have your own group doing physics, I believe it. Maybe even like the one in Milan, with only women;

Viktor, the emphasis you put on te*ll*ing stories is unparalleled;

Giovanni, luckily an Italian will continue taking care of the coffee machine;

Marie, you let me be the coolest guy at the Mainz (Oppenheim) summer school;

Laura, thank you for the runs and the coffees together. Next time it's my turn *to* pay, trust me;

A notable thank you to all the collaborators I have not mentioned yet, David Shih, Florian Ernst, Friedrich Feiden, Jan Rüschkamp, Peter Sorrenson, Ranit Das, Tanmoy Modak.

I would also like to Thank Stella and Luigi from "Divino", who always welcomed us with a "Buongiorno!" and a radiant smile.

Finally, I am grateful to my family, for always showing their support regardless of the distance and the countless times I have not replied to their calls.

From top to bottom, *laughing* with you is what made this possible and I am grateful to have met you all.

It may not have been easy, but in the direst hour, you wrested that very possibility from the depths.

Bibliography

[1] B. M. Dillon, L. Favaro, T. Plehn, P. Sorrenson and M. Krämer, *A normalized autoencoder for LHC triggers*, SciPost Phys. Core **6**, 074 (2023), doi:10.21468/SciPostPhysCore.6.4.074, arXiv:2206.14225.

[2] R. Das, L. Favaro, T. Heimel, C. Krause, T. Plehn and D. Shih, *How to understand limitations of generative networks*, SciPost Phys. **16**, 031 (2024), doi:10.21468/SciPostPhys.16.1.031, arXiv:2305.16774.

[3] B. M. Dillon, L. Favaro, F. Feiden, T. Modak and T. Plehn, *Anomalies, Representations, and Self-Supervision* (2023), arXiv:2301.04660.

[4] L. Favaro, M. Krämer, T. Modak, T. Plehn and J. Rüschkamp, *Semi-visible jets, energy-based models, and self-supervision* (2023), arXiv:2312.03067.

[5] F. Ernst, L. Favaro, C. Krause, T. Plehn and D. Shih, *Normalizing Flows for High-Dimensional Detector Simulations* (2023), arXiv:2312.09290.

[6] L. Favaro, A. Ore, S. P. Schweitzer and T. Plehn, *CaloDREAM – Detector Response Emulation via Attentive flow Matching* (2024), arXiv:2405.09629.

[7] J. M. Campbell *et al.*, *Event generators for high-energy physics experiments*, SciPost Phys. **16**, 130 (2024), doi:10.21468/SciPostPhys.16.5.130, arXiv:2203.11110.

[8] S. Badger *et al.*, *Machine learning and LHC event generation*, SciPost Phys. **14**, 079 (2023), doi:10.21468/SciPostPhys.14.4.079, arXiv:2203.07460.

[9] T. Plehn, A. Butter, B. Dillon, T. Heimel, C. Krause and R. Winterhalder, *Modern Machine Learning for LHC Physicists* (2022), arXiv:2211.01421.

[10] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn, *GANplifying event samples*, SciPost Phys. **10**, 139 (2021), doi:10.21468/SciPostPhys.10.6.139, arXiv:2008.06545.

[11] S. Bieringer, A. Butter, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, B. Nachman, T. Plehn and M. Trabs, *Calomplification — the power of generative calorimeter models*, JINST **17**, P09028 (2022), doi:10.1088/1748-0221/17/09/P09028, arXiv:2202.07352.

[12] M. Paganini, L. de Oliveira and B. Nachman, *Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters*, Phys. Rev. Lett. **120**, 042003 (2018), doi:10.1103/PhysRevLett.120.042003, arXiv:1705.02355.

[13] L. de Oliveira, M. Paganini and B. Nachman, *Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters*, J. Phys. Conf. Ser. **1085**, 042017 (2018), doi:10.1088/1742-6596/1085/4/042017, arXiv:1711.08813.

[14] M. Paganini, L. de Oliveira and B. Nachman, *CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks*, Phys. Rev. D **97**, 014021 (2018), doi:10.1103/PhysRevD.97.014021, arXiv:1712.10321.

[15] M. Erdmann, L. Geiger, J. Glombitza and D. Schmidt, *Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks*, Comput. Softw. Big Sci. **2**, 4 (2018), doi:10.1007/s41781-018-0008-x, arXiv:1802.03325.

[16] M. Erdmann, J. Glombitza and T. Quast, *Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network*, Comput. Softw. Big Sci. **3**, 4 (2019), doi:10.1007/s41781-018-0019-7, arXiv:1807.01954.

[17] D. Belayneh *et al.*, *Calorimetry with deep learning: particle simulation and reconstruction for collider physics*, Eur. Phys. J. C **80**, 688 (2020), doi:10.1140/epjc/s10052-020-8251-9, arXiv:1912.06794.

[18] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger, *Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed*, Comput. Softw. Big Sci. **5**, 13 (2021), doi:10.1007/s41781-021-00056-0, arXiv:2005.05334.

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

[20] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger, *Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network*, EPJ Web Conf. **251**, 03003 (2021), doi:10.1051/epjconf/202125103003, arXiv:2102.12491.

[21] C. Krause and D. Shih, *Fast and accurate simulations of calorimeter showers with normalizing flows*, Phys. Rev. D **107**, 113003 (2023), doi:10.1103/PhysRevD.107.113003, arXiv:2106.05285.

[22] G. Aad *et al.*, *AtlFast3: The Next Generation of Fast Simulation in ATLAS*, Comput. Softw. Big Sci. **6**, 7 (2022), doi:10.1007/s41781-021-00079-7, arXiv:2109.02551.

[23] C. Krause and D. Shih, *Accelerating accurate simulations of calorimeter showers with normalizing flows and probability density distillation*, Phys. Rev. D **107**, 113004 (2023), doi:10.1103/PhysRevD.107.113004, arXiv:2110.11377.

[24] E. Buhmann, S. Diefenbacher, D. Hundhausen, G. Kasieczka, W. Korcari, E. Eren, F. Gaede, K. Krüger, P. McKeown and L. Rustige, *Hadrons, better, faster, stronger*, Mach. Learn. Sci. Tech. **3**, 025014 (2022), doi:10.1088/2632-2153/ac7848, arXiv:2112.09709.

[25] C. Chen, O. Cerri, T. Q. Nguyen, J. R. Vlimant and M. Pierini, *Analysis-Specific Fast Simulation at the LHC with Deep Learning*, Comput. Softw. Big Sci. **5**, 15 (2021), doi:10.1007/s41781-021-00060-4.

[26] A. Adelmann *et al.*, *New directions for surrogate models and differentiable programming for High Energy Physics detector simulation*, In *Snowmass 2021* (2022), arXiv:2203.08806.

[27] V. Mikuni and B. Nachman, *Score-based generative models for calorimeter shower simulation*, Phys. Rev. D **106**, 092009 (2022), doi:10.1103/PhysRevD.106.092009, arXiv:2206.11898.

[28] G. Aad *et al.*, *Deep Generative Models for Fast Photon Shower Simulation in ATLAS*, Comput. Softw. Big Sci. **8**, 7 (2024), doi:10.1007/s41781-023-00106-9, arXiv:2210.06204.

[29] C. Krause, I. Pang and D. Shih, *CaloFlow for CaloChallenge dataset 1*, SciPost Phys. **16**, 126 (2024), doi:10.21468/SciPostPhys.16.5.126, arXiv:2210.14245.

[30] J. C. Cresswell, B. L. Ross, G. Loaiza-Ganem, H. Reyes-Gonzalez, M. Letizia and A. L. Caterini, *CaloMan: Fast generation of calorimeter showers with density estimation on learned manifolds*, In *36th Conference on Neural Information Processing Systems: Workshop on Machine Learning and the Physical Sciences* (2022), arXiv:2211.15380.

[31] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh and D. Shih, *L2LFlows: generating high-fidelity 3D calorimeter images*, JINST **18**, P10017 (2023), doi:10.1088/1748-0221/18/10/P10017, arXiv:2302.11594.

[32] B. Hashemi, N. Hartmann, S. Sharifzadeh, J. Kahn and T. Kuhr, *Ultra-high-granularity detector simulation with intra-event aware generative adversarial network and self-supervised relational reasoning*, Nature Commun. **15**, 4916 (2024), doi:10.1038/s41467-024-49104-4, [Erratum: Nature Commun. 115, 5825 (2024)], arXiv:2303.08046.

[33] A. Xu, S. Han, X. Ju and H. Wang, *Generative machine learning for detector response modeling with a conditional normalizing flow*, JINST **19**, P02003 (2024), doi:10.1088/1748-0221/19/02/P02003, arXiv:2303.10148.

[34] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, K. Krüger, P. McKeown and L. Rustige, *New angles on fast calorimeter shower simulation*, Mach. Learn. Sci. Tech. **4**, 035044 (2023), doi:10.1088/2632-2153/acefa9, arXiv:2303.18150.

[35] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown, *CaloClouds: fast geometry-independent highly-granular calorimeter simulation*, JINST **18**, P11025 (2023), doi:10.1088/1748-0221/18/11/P11025, arXiv:2305.04847.

[36] M. R. Buckley, C. Krause, I. Pang and D. Shih, *Inductive simulation of calorimeter showers with normalizing flows*, Phys. Rev. D **109**, 033006 (2024), doi:10.1103/PhysRevD.109.033006, arXiv:2305.11934.

[37] V. Mikuni and B. Nachman, *CaloScore v2: single-shot calorimeter shower simulation with diffusion models*, JINST **19**, P02001 (2024), doi:10.1088/1748-0221/19/02/P02001, arXiv:2308.03847.

[38] O. Amram and K. Pedro, *Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation*, Phys. Rev. D **108**, 072014 (2023), doi:10.1103/PhysRevD.108.072014, arXiv:2308.03876.

[39] S. Diefenbacher, V. Mikuni and B. Nachman, *Refining Fast Calorimeter Simulations with a Schrödinger Bridge* (2023), arXiv:2308.12339.

[40] M. Faucci Giannelli and R. Zhang, *CaloShowerGAN, a generative adversarial network model for fast calorimeter shower simulation*, Eur. Phys. J. Plus **139**, 597 (2024), doi:10.1140/epjp/s13360-024-05397-4, arXiv:2309.06515.

[41] I. Pang, D. Shih and J. A. Raine, *Calorimeter shower superresolution*, Phys. Rev. D **109**, 092009 (2024), doi:10.1103/PhysRevD.109.092009, arXiv:2308.11700.

[42] S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman and D. Shih, *DCTRGAN: Improving the Precision of Generative Models with Reweighting*, JINST **15**, P11004 (2020), doi:10.1088/1748-0221/15/11/P11004, arXiv:2009.03796.

[43] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot and S. Vent, *Generative networks for precision enthusiasts*, SciPost Phys. **14**, 078 (2023), doi:10.21468/SciPostPhys.14.4.078, arXiv:2110.13632.

[44] R. Winterhalder, M. Bellagente and B. Nachman, *Latent Space Refinement for Deep Generative Models* (2021), arXiv:2106.00792.

[45] B. Nachman and R. Winterhalder, *Elsa: enhanced latent spaces for improved collider simulations*, Eur. Phys. J. C **83**, 843 (2023), doi:10.1140/epjc/s10052-023-11989-8, arXiv:2305.07696.

[46] M. Leigh, D. Sengupta, J. A. Raine, G. Quétant and T. Golling, *Faster diffusion model with improved quality for particle cloud generation*, Phys. Rev. D **109**, 012010 (2024), doi:10.1103/PhysRevD.109.012010, arXiv:2307.06836.

[47] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear Independent Components Estimation*, arXiv e-prints arXiv:1410.8516 (2014), arXiv:1410.8516.

[48] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density estimation using Real NVP*, arXiv e-prints arXiv:1605.08803 (2016), arXiv:1605.08803.

[49] D. P. Kingma and P. Dhariwal, *Glow: Generative flow with invertible 1x1 convolutions*, In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, eds., *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc. (2018), arXiv:1807.03039.

[50] J. A. Aguilar-Saavedra, J. H. Collins and R. K. Mishra, *A generic anti-QCD jet tagger*, JHEP **11**, 163 (2017), doi:10.1007/JHEP11(2017)163, arXiv:1709.01087.

[51] T. S. Roy and A. H. Vijay, *A robust anomaly finder based on autoencoders* (2019), arXiv:1903.02032.

[52] B. M. Dillon, D. A. Faroughy and J. F. Kamenik, *Uncovering latent jet substructure*, Phys. Rev. D **100**, 056002 (2019), doi:10.1103/PhysRevD.100.056002, arXiv:1904.04200.

[53] J. A. Aguilar-Saavedra, F. R. Joaquim and J. F. Seabra, *Mass Unspecific Supervised Tagging (MUST) for boosted jets*, JHEP **03**, 012 (2021), doi:10.1007/JHEP03(2021)012, [Erratum: JHEP 04, 133 (2021)], arXiv:2008.12792.

[54] O. Atkinson, A. Bhardwaj, C. Englert, V. S. Ngairangbam and M. Spannowsky, *Anomaly detection with convolutional Graph Neural Networks*, JHEP **08**, 080 (2021), doi:10.1007/JHEP08(2021)080, arXiv:2105.07988.

[55] A. Kahn, J. Gonski, I. Ochoa, D. Williams and G. Brooijmans, *Anomalous jet identification via sequence modeling*, JINST **16**, P08012 (2021), doi:10.1088/1748-0221/16/08/P08012, arXiv:2105.09274.

[56] M. J. Dolan and A. Ore, *Metalearning and data augmentation for mass-generalized jet taggers*, Phys. Rev. D **105**, 094030 (2022), doi:10.1103/PhysRevD.105.094030, arXiv:2111.06047.

[57] F. Canelli, A. de Cosa, L. L. Pottier, J. Niedziela, K. Pedro and M. Pierini, *Autoencoders for semivisible jet detection*, JHEP **02**, 074 (2022), doi:10.1007/JHEP02(2022)074, arXiv:2112.02864.

[58] T. Buss, B. M. Dillon, T. Finke, M. Krämer, A. Morandini, A. Mück, I. Oleksiyuk and T. Plehn, *What's anomalous in LHC jets?*, SciPost Phys. **15**, 168 (2023), doi:10.21468/SciPostPhys.15.4.168, arXiv:2202.00686.

[59] O. Atkinson, A. Bhardwaj, C. Englert, P. Konar, V. S. Ngairangbam and M. Spannowsky, *IRC-Safe Graph Autoencoder for Unsupervised Anomaly Detection*, Front. Artif. Intell. **5**, 943135 (2022), doi:10.3389/frai.2022.943135, arXiv:2204.12231.

[60] R. T. D'Agnolo and A. Wulzer, *Learning New Physics from a Machine*, Phys. Rev. D **99**, 015014 (2019), doi:10.1103/PhysRevD.99.015014, arXiv:1806.02350.

[61] J. Hajer, Y.-Y. Li, T. Liu and H. Wang, *Novelty Detection Meets Collider Physics*, Phys. Rev. D **101**, 076015 (2020), doi:10.1103/PhysRevD.101.076015, arXiv:1807.10261.

[62] P. T. Komiske, E. M. Metodiev and J. Thaler, *Metric Space of Collider Events*, Phys. Rev. Lett. **123**, 041801 (2019), doi:10.1103/PhysRevLett.123.041801, arXiv:1902.02346.

[63] A. Blance, M. Spannowsky and P. Waite, *Adversarially-trained autoencoders for robust unsupervised new physics searches*, JHEP **10**, 047 (2019), doi:10.1007/JHEP10(2019)047, arXiv:1905.10384.

[64] M. Romão Crispim, N. F. Castro, R. Pedro and T. Vale, *Transferability of Deep Learning Models in Searches for New Physics at Colliders*, Phys. Rev. D **101**, 035042 (2020), doi:10.1103/PhysRevD.101.035042, arXiv:1912.04220.

[65] R. T. D'Agnolo, G. Grosso, M. Pierini, A. Wulzer and M. Zanetti, *Learning multivariate new physics*, Eur. Phys. J. C **81**, 89 (2021), doi:10.1140/epjc/s10052-021-08853-y, arXiv:1912.12155.

[66] A. Andreassen, B. Nachman and D. Shih, *Simulation Assisted Likelihood-free Anomaly Detection*, Phys. Rev. D **101**, 095004 (2020), doi:10.1103/PhysRevD.101.095004, arXiv:2001.05001.

[67] O. Amram and C. M. Suarez, *Tag N' Train: a technique to train improved classifiers on unlabeled data*, JHEP **01**, 153 (2021), doi:10.1007/JHEP01(2021)153, arXiv:2002.12376.

[68] K. T. Matchev, P. Shyamsundar and J. Smolinsky, *A Quantum Algorithm for Model-Independent Searches for New Physics*, LHEP **2023**, 301 (2023), doi:10.31526/l-hep.2023.301, arXiv:2003.02181.

[69] P. T. Komiske, E. M. Metodiev and J. Thaler, *The Hidden Geometry of Particle Collisions*, JHEP **07**, 006 (2020), doi:10.1007/JHEP07(2020)006, arXiv:2004.04159.

[70] M. Crispim Romão, N. F. Castro, J. G. Milhano, R. Pedro and T. Vale, *Use of a generalized energy Mover's distance in the search for rare phenomena at colliders*, Eur. Phys. J. C **81**, 192 (2021), doi:10.1140/epjc/s10052-021-08891-6, arXiv:2004.09360.

[71] B. M. Dillon, D. A. Faroughy, J. F. Kamenik and M. Szewc, *Learning the latent structure of collider events*, JHEP **10**, 206 (2020), doi:10.1007/JHEP10(2020)206, arXiv:2005.12319.

[72] M. Crispim Romão, N. F. Castro and R. Pedro, *Finding New Physics without learning about it: Anomaly Detection as a tool for Searches at Colliders*, Eur. Phys. J. C **81**, 27 (2021), doi:10.1140/epjc/s10052-021-09813-2, [Erratum: Eur.Phys.J.C 81, 1020 (2021)], arXiv:2006.05432.

[73] C. K. Khosa and V. Sanz, *Anomaly Awareness*, SciPost Phys. **15**, 053 (2023), doi:10.21468/SciPostPhys.15.2.053, arXiv:2007.14462.

[74] V. Mikuni and F. Canelli, *Unsupervised clustering for collider physics*, Phys. Rev. D **103**, 092007 (2021), doi:10.1103/PhysRevD.103.092007, arXiv:2010.07106.

[75] M. van Beekveld, S. Caron, L. Hendriks, P. Jackson, A. Leinweber, S. Otten, R. Patrick, R. Ruiz De Austri, M. Santoni and M. White, *Combining outlier analysis algorithms to identify new physics at the LHC*, JHEP **09**, 024 (2021), doi:10.1007/JHEP09(2021)024, arXiv:2010.07940.

[76] P. Jawahar, T. Aarrestad, N. Chernyavskaya, M. Pierini, K. A. Wozniak, J. Ngadiuba, J. Duarte and S. Tsan, *Improving Variational Autoencoders for New Physics Detection at the LHC With Normalizing Flows*, Front. Big Data **5**, 803685 (2022), doi:10.3389/fdata.2022.803685, arXiv:2110.08508.

[77] V. Mikuni, B. Nachman and D. Shih, *Online-compatible unsupervised nonresonant anomaly detection*, Phys. Rev. D **105**, 055006 (2022), doi:10.1103/PhysRevD.105.055006, arXiv:2111.06417.

[78] L. Bradshaw, S. Chang and B. Ostdiek, *Creating simple, interpretable anomaly detectors for new physics in jet substructure*, Phys. Rev. D **106**, 035014 (2022), doi:10.1103/PhysRevD.106.035014, arXiv:2203.01343.

[79] J. H. Collins, K. Howe and B. Nachman, *Anomaly Detection for Resonant New Physics with Machine Learning*, Phys. Rev. Lett. **121**, 241803 (2018), doi:10.1103/PhysRevLett.121.241803, arXiv:1805.02664.

[80] A. De Simone and T. Jacques, *Guiding New Physics Searches with Unsupervised Learning*, Eur. Phys. J. C **79**, 289 (2019), doi:10.1140/epjc/s10052-019-6787-3, arXiv:1807.06038.

[81] J. H. Collins, K. Howe and B. Nachman, *Extending the search for new resonances with machine learning*, Phys. Rev. D **99**, 014038 (2019), doi:10.1103/PhysRevD.99.014038, arXiv:1902.02634.

[82] A. Mullin, S. Nicholls, H. Pacey, M. Parker, M. White and S. Williams, *Does SUSY have friends? A new approach for LHC event analysis*, JHEP **02**, 160 (2021), doi:10.1007/JHEP02(2021)160, arXiv:1912.10625.

[83] K. Benkendorfer, L. L. Pottier and B. Nachman, *Simulation-assisted decorrelation for resonant anomaly detection*, Phys. Rev. D **104**, 035003 (2021), doi:10.1103/PhysRevD.104.035003, arXiv:2009.02205.

[84] S. E. Park, D. Rankin, S.-M. Udrescu, M. Yunus and P. Harris, *Quasi Anomalous Knowledge: Searching for new physics with embedded knowledge*, JHEP **21**, 030 (2020), doi:10.1007/JHEP06(2021)030, arXiv:2011.03550.

[85] A. Hallin, J. Isaacson, G. Kasieczka, C. Krause, B. Nachman, T. Quadfasel, M. Schlaffer, D. Shih and M. Sommerhalder, *Classifying anomalies through outer density estimation*, Phys. Rev. D **106**, 055006 (2022), doi:10.1103/PhysRevD.106.055006, arXiv:2109.00546.

[86] J. Barron, D. Curtin, G. Kasieczka, T. Plehn and A. Spourdalakis, *Unsupervised hadronic SUEP at the LHC*, JHEP **12**, 129 (2021), doi:10.1007/JHEP12(2021)129, arXiv:2107.12379.

[87] T. Finke, M. Krämer, M. Lipp and A. Mück, *Boosting mono-jet searches with model-agnostic machine learning*, JHEP **08**, 015 (2022), doi:10.1007/JHEP08(2022)015, arXiv:2204.11889.

[88] G. Aad *et al.*, *Dijet resonance search with weak supervision using $\sqrt{s} = 13$ TeV pp collisions in the ATLAS detector*, Phys. Rev. Lett. **125**, 131801 (2020), doi:10.1103/PhysRevLett.125.131801, arXiv:2005.02983.

[89] P. T. Komiske, R. Mastandrea, E. M. Metodiev, P. Naik and J. Thaler, *Exploring the Space of Jets with CMS Open Data*, Phys. Rev. D **101**, 034009 (2020), doi:10.1103/PhysRevD.101.034009, arXiv:1908.08542.

[90] O. Knapp, O. Cerri, G. Dissertori, T. Q. Nguyen, M. Pierini and J.-R. Vlimant, *Adversarially Learned Anomaly Detection on CMS Open Data: re-discovering the top quark*, Eur. Phys. J. Plus **136**, 236 (2021), doi:10.1140/epjp/s13360-021-01109-4, arXiv:2005.01598.

[91] P. Thaprasop, K. Zhou, J. Steinheimer and C. Herold, *Unsupervised Outlier Detection in Heavy-Ion Collisions*, Phys. Scripta **96**, 064003 (2021), doi:10.1088/1402-4896/abf214, arXiv:2007.15830.

[92] T. Aarrestad *et al.*, *The Dark Machines Anomaly Score Challenge: Benchmark Data and Model Independent Event Classification for the Large Hadron Collider*, SciPost Phys. **12**, 043 (2022), doi:10.21468/SciPostPhys.12.1.043, arXiv:2105.14027.

[93] G. Kasieczka *et al.*, *The LHC Olympics 2020 a community challenge for anomaly detection in high energy physics*, Rept. Prog. Phys. **84**, 124201 (2021), doi:10.1088/1361-6633/ac36b9, arXiv:2101.08320.

[94] B. Bortolato, A. Smolkovič, B. M. Dillon and J. F. Kamenik, *Bump hunting in latent space*, Phys. Rev. D **105**, 115009 (2022), doi:10.1103/PhysRevD.105.115009, arXiv:2103.06595.

[95] B. Nachman and D. Shih, *Anomaly Detection with Density Estimation*, Phys. Rev. D **101**, 075042 (2020), doi:10.1103/PhysRevD.101.075042, arXiv:2001.04990.

[96] J. Batson, C. G. Haaf, Y. Kahn and D. A. Roberts, *Topological Obstructions to Autoencoding*, JHEP **04**, 280 (2021), doi:10.1007/JHEP04(2021)280, arXiv:2102.08380.

[97] T. Dorigo, M. Fumanelli, C. Maccani, M. Mojsovska, G. C. Strong and B. Scarpa, *RanBox: anomaly detection in the copula space*, JHEP **01**, 008 (2023), doi:10.1007/JHEP01(2023)008, arXiv:2106.05747.

[98] S. Caron, L. Hendriks and R. Verheyen, *Rare and Different: Anomaly Scores from a combination of likelihood and out-of-distribution models to detect new physics at the LHC*, SciPost Phys. **12**, 077 (2022), doi:10.21468/SciPostPhys.12.2.077, arXiv:2106.10164.

[99] K. Fraser, S. Homiller, R. K. Mishra, B. Ostdiek and M. D. Schwartz, *Challenges for unsupervised anomaly detection in particle physics*, JHEP **03**, 066 (2022), doi:10.1007/JHEP03(2022)066, arXiv:2110.06948.

[100] G. Kasieczka, R. Mastandrea, V. Mikuni, B. Nachman, M. Pettee and D. Shih, *Anomaly detection under coordinate transformations*, Phys. Rev. D **107**, 015009 (2023), doi:10.1103/PhysRevD.107.015009, arXiv:2209.06225.

[101] T. Chen, S. Kornblith, M. Norouzi and G. E. Hinton, *A Simple Framework for Contrastive Learning of Visual Representations*, Proceedings of the 37th International Conference on Machine Learning, PMLR **119**, 1597 (2020), `https://proceedings.mlr.press/v119/chen20j.html`, arXiv:2002.05709.

[102] S. Weinberg, *A Model of Leptons*, Phys. Rev. Lett. **19**, 1264 (1967), doi:10.1103/PhysRevLett.19.1264.

[103] S. L. Glashow, *Partial Symmetries of Weak Interactions*, Nucl. Phys. **22**, 579 (1961), doi:10.1016/0029-5582(61)90469-2.

[104] A. Salam, *Weak and Electromagnetic Interactions*, Conf. Proc. C **680519**, 367 (1968), doi:10.1142/9789812795915_0034.

[105] M. E. Peskin and D. V. Schroeder, *An Introduction to quantum field theory*, Addison-Wesley, Reading, USA, ISBN 978-0-201-50397-5, 978-0-429-50355-9, 978-0-429-49417-8, doi:10.1201/9780429503559 (1995).

[106] *Standard Model Summary Plots October 2023* (2023).

[107] J. Alwall *et al.*, *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, JHEP **07**, 079 (2014), doi:10.1007/JHEP07(2014)079, arXiv:1405.0301.

[108] C. Bierlich *et al.*, *A comprehensive guide to the physics and usage of PYTHIA 8.3*, SciPost Phys. Codeb. **2022**, 8 (2022), doi:10.21468/SciPostPhysCodeb.8, arXiv:2203.11601.

[109] E. Bothmann *et al.*, *Event Generation with Sherpa 2.2*, SciPost Phys. **7**, 034 (2019), doi:10.21468/SciPostPhys.7.3.034, arXiv:1905.09127.

[110] R. D. Ball *et al.*, *The path to proton structure at 1% accuracy*, Eur. Phys. J. C **82**, 428 (2022), doi:10.1140/epjc/s10052-022-10328-7, arXiv:2109.02653.

[111] M. D. Schwartz, *Quantum Field Theory and the Standard Model*, Cambridge University Press, ISBN 978-1-107-03473-0, 978-1-107-03473-0 (2014).

[112] G. Bewick *et al.*, *Herwig 7.3 Release Note* (2023), arXiv:2312.05175.

[113] B. Andersson, G. Gustafson, G. Ingelman and T. Sjostrand, *Parton Fragmentation and String Dynamics*, Phys. Rept. **97**, 31 (1983), doi:10.1016/0370-1573(83)90080-7.

[114] J.-C. Winter, F. Krauss and G. Soff, *A Modified cluster hadronization model*, Eur. Phys. J. C **36**, 381 (2004), doi:10.1140/epjc/s2004-01960-8, arXiv:hep-ph/0311085.

[115] R. L. Workman *et al.*, *Review of Particle Physics*, PTEP **2022**, 083C01 (2022), doi:10.1093/ptep/ptac097.

[116] J. D. Cockcroft, *Experimental nuclear physics*, Nature **175**, 53 (1955), `https://api.semanticscholar.org/CorpusID:4249052`.

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

[118] J. Allison *et al.*, *Geant4 developments and applications*, IEEE Transactions on Nuclear Science **53**, 270 (2006), doi:10.1109/TNS.2006.869826.

[119] J. Allison *et al.*, *Recent developments in geant4*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **835**, 186 (2016), doi:https://doi.org/10.1016/j.nima.2016.06.125, `https://www.sciencedirect.com/science/article/pii/S0168900216306957`.

[120] Y.-S. Tsai, *Pair production and bremsstrahlung of charged leptons*, Rev. Mod. Phys. **46**, 815 (1974), doi:10.1103/RevModPhys.46.815, `https://link.aps.org/doi/10.1103/RevModPhys.46.815`.

[121] A. M. Sirunyan *et al.*, *Particle-flow reconstruction and global event description with the CMS detector*, JINST **12**, P10003 (2017), doi:10.1088/1748-0221/12/10/P10003, arXiv:1706.04965.

[122] M. Aaboud *et al.*, *Jet reconstruction and performance using particle flow with the ATLAS Detector*, Eur. Phys. J. C **77**, 466 (2017), doi:10.1140/epjc/s10052-017-5031-2, arXiv:1703.10485.

[123] G. Aad *et al.*, *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 (2012), doi:10.1016/j.physletb.2012.08.020, arXiv:1207.7214.

[124] S. Chatrchyan *et al.*, *Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC*, Phys. Lett. B **716**, 30 (2012), doi:10.1016/j.physletb.2012.08.021, arXiv:1207.7235.

[125] H. Qu, C. Li and S. Qian, *Particle Transformer for Jet Tagging* (2022), arXiv:2202.03772.

[126] A. Bogatskiy, T. Hoffman, D. W. Miller and J. T. Offermann, *PELICAN: Permutation Equivariant and Lorentz Invariant or Covariant Aggregator Network for Particle Physics* (2022), arXiv:2211.00454.

[127] G. Aad *et al.*, *Search for New Phenomena in Two-Body Invariant Mass Distributions Using Unsupervised Machine Learning for Anomaly Detection at s=13 TeV with the ATLAS Detector*, Phys. Rev. Lett. **132**, 081801 (2024), doi:10.1103/PhysRevLett.132.081801, arXiv:2307.01612.

[128] *Model-agnostic search for dijet resonances with anomalous jet substructure in proton-proton collisions at $\sqrt{s}$ = 13 TeV* (2024).

[129] A. Butter, N. Huetsch, S. Palacios Schweitzer, T. Plehn, P. Sorrenson and J. Spinner, *Jet Diffusion versus JetGPT – Modern Networks for the LHC* (2023), arXiv:2305.10475.

[130] G. V. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals and Systems **2**, 303 (1989), `https://api.semanticscholar.org/CorpusID:3958369`.

[131] M. Leshno, V. Y. Lin, A. Pinkus and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Networks **6**, 861 (1993), doi:https://doi.org/10.1016/S0893-6080(05)80131-5.

[132] R. Rojas, *The Backpropagation Algorithm*, pp. 149–182, Springer Berlin Heidelberg, Berlin, Heidelberg (1996).

[133] L. N. Smith and N. Topin, *Super-convergence: Very fast training of neural networks using large learning rates*, In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006, p. 1100612. International Society for Optics and Photonics (2019).

[134] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization* (2014), arXiv:1412.6980.

[135] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization* (2019), arXiv:1711.05101.

[136] J. Neyman and E. S. Pearson, *On the Problem of the Most Efficient Tests of Statistical Hypotheses*, Phil. Trans. Roy. Soc. Lond. A **231**, 289 (1933), doi:10.1098/rsta.1933.0009.

[137] S. Rizvi, M. Pettee and B. Nachman, *Learning likelihood ratios with neural network classifiers*, JHEP **02**, 136 (2024), doi:10.1007/JHEP02(2024)136, arXiv:2305.10500.

[138] Y. Song and D. P. Kingma, *How to train your energy-based models*, doi:10.48550/ARXIV.2101.03288 (2021).

[139] T. Heimel, G. Kasieczka, T. Plehn and J. M. Thompson, *QCD or What?*, SciPost Phys. **6**, 030 (2019), doi:10.21468/SciPostPhys.6.3.030, arXiv:1808.08979.

[140] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed and A. Lerchner, *beta-VAE: Learning basic visual concepts with a constrained variational framework*, In *International Conference on Learning Representations* (2017).

[141] A. Bogatskiy, B. Anderson, J. T. Offermann, M. Roussi, D. W. Miller and R. Kondor, *Lorentz Group Equivariant Neural Network for Particle Physics* (2020), arXiv:2006.04780.

[142] J. Spinner, V. Bresó, P. de Haan, T. Plehn, J. Thaler and J. Brehmer, *Lorentz-Equivariant Geometric Algebra Transformers for High-Energy Physics* (2024), arXiv:2405.14806.

[143] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother and U. Köthe, *Analyzing inverse problems with invertible neural networks* (2018), arXiv:1808.04730.

[144] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density estimation using real nvp* (2016), arXiv:1605.08803.

[145] T. Heimel, R. Winterhalder, A. Butter, J. Isaacson, C. Krause, F. Maltoni, O. Mattelaer and T. Plehn, *MadNIS - Neural multi-channel importance sampling*, SciPost Phys. **15**, 141 (2023), doi:10.21468/SciPostPhys.15.4.141, arXiv:2212.06172.

[146] G. Papamakarios, T. Pavlakou and I. Murray, *Masked autoregressive flow for density estimation*, Advances in neural information processing systems **30** (2017), 1705.07057.

[147] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever and M. Welling, *Improved variational inference with inverse autoregressive flow*, Advances in neural information processing systems **29** (2016).

[148] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Neural spline flows*, Advances in Neural Information Processing Systems **32**, 7511 (2019), 1906.04032.

[149] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Cubic-spline flows*, arXiv preprint arXiv:1906.02145 (2019), 1906.02145.

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

[151] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 - dataset 1*, `https://doi.org/10.5281/zenodo.6234054` (2022).

[152] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 - dataset 1 version 3*, `https://doi.org/10.5281/zenodo.8099322` (2023).

[153] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 - dataset 2*, `https://doi.org/10.5281/zenodo.6366271` (2022).

[154] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 - dataset 3*, `https://doi.org/10.5281/zenodo.6366324` (2022).

[155] M. Faucci Giannelli, G. Kasieczka, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 github page*, `https://github.com/CaloChallenge/homepage` (2022).

[156] M. F. Giannelli, *private communication* (2023).

[157] R. Kansal, A. Li, J. Duarte, N. Chernyavskaya, M. Pierini, B. Orzari and T. Tomei, *Evaluating generative models in high energy physics*, Phys. Rev. D **107**, 076017 (2023), doi:10.1103/PhysRevD.107.076017, arXiv:2211.10295.

[158] S. Badger, A. Butter, M. Luchmann, S. Pitz and T. Plehn, *Loop amplitudes from precision networks*, SciPost Phys. Core **6**, 034 (2023), doi:10.21468/SciPostPhysCore.6.2.034, arXiv:2206.14831.

[159] T. Faucett, J. Thaler and D. Whiteson, *Mapping Machine-Learned Physics into a Human-Readable Space*, Phys. Rev. D **103**, 036020 (2021), doi:10.1103/PhysRevD.103.036020, arXiv:2010.11998.

[160] R. Das, G. Kasieczka and D. Shih, *Feature selection with distance correlation*, Phys. Rev. D **109**, 054009 (2024), doi:10.1103/PhysRevD.109.054009, arXiv:2212.00046.

[161] L. Ardizzone, C. Lüth, J. Kruse, C. Rother and U. Köthe, *Guided image generation with conditional invertible neural networks* (2019), arXiv:1907.02392.

[162] L. Ardizzone, T. Bungert, F. Draxler, U. Köthe, J. Kruse, R. Schmier and P. Sorrenson, *Framework for Easily Invertible Architectures (FrEIA)* (2018-2022).

[163] G. Loaiza-Ganem and J. P. Cunningham, *The continuous bernoulli: fixing a pervasive error in variational autoencoders* (2019), arXiv:1907.06845.

[164] G. M. Kurtzer, V. Sochat and M. W. Bauer, *Singularity: Scientific containers for mobility of compute*, PLOS ONE **12**, 1 (2017), doi:10.1371/journal.pone.0177459, `https://doi.org/10.1371/journal.pone.0177459`.

[165] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel and M. Le, *Flow matching for generative modeling,* arXiv preprint arXiv:2210.02747 (2022).

[166] T. Heimel, N. Huetsch, R. Winterhalder, T. Plehn and A. Butter, *Precision-Machine Learning for the Matrix Element Method* (2023), arXiv:2310.07752.

[167] W. S. Peebles and S. Xie, *Scalable diffusion models with transformers*, 2023 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 4172–4182 (2022), `https://api.semanticscholar.org/CorpusID:254854389`.

[168] M. Bellagente, M. Haussmann, M. Luchmann and T. Plehn, *Understanding Event-Generation Networks via Uncertainties*, SciPost Phys. **13**, 003 (2022), doi:10.21468/SciPostPhys.13.1.003, arXiv:2104.04543.

[169] Q. Liu, C. Shimmin, X. Liu, E. Shlizerman, S. Li and S.-C. Hsu, *Calo-VQ: Vector-Quantized Two-Stage Generative Model in Calorimeter Simulation* (2024), arXiv:2405.06605.

[170] T. Salimans and J. Ho, *Progressive distillation for fast sampling of diffusion models*, In *International Conference on Learning Representations* (2022).

[171] Y. Song, P. Dhariwal, M. Chen and I. Sutskever, *Consistency models*, In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato and J. Scarlett, eds., *Proceedings of the 40th International Conference on Machine Learning*, vol. 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR (2023).

[172] E. Buhmann, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown, *CaloClouds II: ultra-fast geometry-independent highly-granular calorimeter simulation*, JINST **19**, P04020 (2024), doi:10.1088/1748-0221/19/04/P04020, arXiv:2309.05704.

[173] C. Jiang, S. Qian and H. Qu, *Choose Your Diffusion: Efficient and flexible ways to accelerate the diffusion model in fast high energy physics simulation* (2024), arXiv:2401.13162.

[174] N. Shaul, J. C. Perez, R. T. Q. Chen, A. K. Thabet, A. Pumarola and Y. Lipman, *Bespoke solvers for generative flow models*, ArXiv **abs/2310.19075** (2023), `https://api.semanticscholar.org/CorpusID:264825556`.

[175] N. Shaul, U. Singer, R. T. Q. Chen, M. Le, A. Thabet, A. Pumarola and Y. Lipman, *Bespoke non-stationary solvers for fast sampling of diffusion and flow models*, ArXiv **abs/2403.01329** (2024), `https://api.semanticscholar.org/CorpusID:268231006`.

[176] R. Wigmans and M. T. Zeyrek, *On the differences between calorimetric detection of electrons and photons*, Nucl. Instrum. Meth. A **485**, 385 (2002), doi:10.1016/S0168-9002(01)02141-6.

[177] A. Butter, G. Kasieczka, T. Plehn and M. Russell, *Deep-learned Top Tagging with a Lorentz Layer*, SciPost Phys. **5**, 028 (2018), doi:10.21468/SciPostPhys.5.3.028, arXiv:1707.08966.

[178] A. Butter *et al.*, *The Machine Learning landscape of top taggers*, SciPost Phys. **7**, 014 (2019), doi:10.21468/SciPostPhys.7.1.014, arXiv:1902.09914.

[179] L. Benato *et al.*, *Shared Data and Algorithms for Deep Learning in Fundamental Physics*, Comput. Softw. Big Sci. **6**, 9 (2022), doi:10.1007/s41781-022-00082-6, arXiv:2107.00656.

[180] B. M. Dillon, T. Plehn, C. Sauer and P. Sorrenson, *Better Latent Spaces for Better Autoencoders*, SciPost Phys. **11**, 061 (2021), doi:10.21468/SciPostPhys.11.3.061, arXiv:2104.08291.

[181] M. Cacciari, G. P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, JHEP **04**, 063 (2008), doi:10.1088/1126-6708/2008/04/063, arXiv:0802.1189.

[182] M. Cacciari, G. P. Salam and G. Soyez, *FastJet User Manual*, Eur. Phys. J. **C72**, 1896 (2012), doi:10.1140/epjc/s10052-012-1896-2, arXiv:1111.6097.

[183] E. Bernreuther, T. Finke, F. Kahlhoefer, M. Krämer and A. Mück, *Casting a graph net to catch dark showers*, SciPost Phys. **10**, 046 (2021), doi:10.21468/SciPostPhys.10.2.046, arXiv:2006.08639.

[184] T. Finke, M. Krämer, A. Morandini, A. Mück and I. Oleksiyuk, *Autoencoders for unsupervised anomaly detection in high energy physics*, JHEP **06**, 161 (2021), doi:10.1007/JHEP06(2021)161, arXiv:2104.09051.

[185] L. Carloni and T. Sjostrand, *Visible Effects of Invisible Hidden Valley Radiation*, JHEP **09**, 105 (2010), doi:10.1007/JHEP09(2010)105, arXiv:1006.2911.

[186] L. Carloni, J. Rathsman and T. Sjostrand, *Discerning Secluded Sector gauge structures*, JHEP **04**, 091 (2011), doi:10.1007/JHEP04(2011)091, arXiv:1102.3795.

[187] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli and M. Zaro, *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, JHEP **07**, 079 (2014), doi:10.1007/JHEP07(2014)079, arXiv:1405.0301.

[188] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen and P. Z. Skands, *An introduction to PYTHIA 8.2*, Comput. Phys. Commun. **191**, 159 (2015), doi:10.1016/j.cpc.2015.01.024, arXiv:1410.3012.

[189] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens and M. Selvaggi, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, JHEP **02**, 057 (2014), doi:10.1007/JHEP02(2014)057, arXiv:1307.6346.

[190] M. Cacciari, G. P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, JHEP **04**, 063 (2008), doi:10.1088/1126-6708/2008/04/063, arXiv:0802.1189.

[191] E. Govorkova, E. Puljak, T. Aarrestad, M. Pierini, K. A. Woźniak and J. Ngadiuba, *LHC physics dataset for unsupervised New Physics detection at 40 MHz*, Sci. Data **9**, 118 (2022), doi:10.1038/s41597-022-01187-8, arXiv:2107.02157.

[192] O. Cerri, T. Q. Nguyen, M. Pierini, M. Spiropulu and J.-R. Vlimant, *Variational Autoencoders for New Physics Mining at the Large Hadron Collider*, JHEP **05**, 036 (2019), doi:10.1007/JHEP05(2019)036, arXiv:1811.10276.

[193] S. Yoon, Y.-K. Noh and F. C. Park, *Autoencoding under normalization constraints*, ICML (2021), arXiv:2105.05735.

[194] Y. Du and I. Mordatch, *Implicit generation and generalization in energy-based models*, doi:10.48550/ARXIV.1903.08689 (2019).

[195] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi and K. Swersky, *Your classifier is secretly an energy based model and you should treat it like one* (2019), doi:10.48550/ARXIV.1912.03263, `https://arxiv.org/abs/1912.03263`.

[196] E. Nijkamp, M. Hill, S.-C. Zhu and Y. N. Wu, *Learning non-convergent non-persistent short-run mcmc toward energy-based model*, Advances in Neural Information Processing Systems **32** (2019).

[197] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf and A. Smola, *A kernel two-sample test*, The Journal of Machine Learning Research **13**, 723 (2012).

[198] B. M. Dillon, G. Kasieczka, H. Olischlager, T. Plehn, P. Sorrenson and L. Vogel, *Symmetries, safety, and self-supervision*, SciPost Phys. **12**, 188 (2022), doi:10.21468/SciPostPhys.12.6.188, arXiv:2108.04253.

[199] G. E. Hinton, *Training products of experts by minimizing contrastive divergence*, Neural Computation **14**, 1771 (2002), doi:10.1162/089976602760128018.

[200] S. Lyu, *Unifying non-maximum likelihood learning objectives with minimum kl contraction*, In *Neural Information Processing Systems* (2011).

[201] M. Sharma, S. Farquhar, E. Nalisnick and T. Rainforth, *Do bayesian neural networks need to be fully stochastic?* (2023), arXiv:2211.06291.