Precision Machine Learning for the LHC Simulation Chain

Dissertation

Sofia Palacios Schweitzer

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

Sofia Palacios Schweitzer

born in Berlin, Germany

Oral examination: 02.07.2025

Precision Machine Learning for the LHC Simulation Chain

Referees:	Dr. Anja Butter
	Prof. Dr. Björn Malte Schäfer

Abstract

The simulation chain of LHC physics is a well-established and indispensable toolkit for conducting precision measurements at general-purpose detectors at the LHC. While not all components are derived from first-principle physics, the simulation chain as a whole has demonstrated remarkable accuracy and reliability over the past decade. To keep pace with increasing experimental demands and growing data statistics, the simulation chain undergoes continuous refinement. In recent years, the rise of machine learning has opened new avenues for further advancing the simulation and analysis pipeline of high energy physics. In this work, we explore the integration of state-of-the-art generative machine learning algorithms into different stages of the LHC simulation chain. First, we test their ability to enhance forward simulations by improving the generation speed, particularly in computationally intensive steps. Second, we apply generative machine learning models to the inverse problem of unfolding detector effects, offering an alternative to traditional techniques. We show that both tasks can be solved using machine learning with high precision and accuracy, demonstrating the potential of these approaches to significantly improve the scalability and robustness of future LHC analyses.

Zusammenfassung

Die LHC-Simulationskette ist ein etabliertes und unverzichtbares Werkzeug für präzise Messungen an den Allzweckdetektoren des LHCs. Obwohl nicht alle Bestandteile auf fundamentaler Physik basieren, hat sich die Kette als Ganzes in den letzten zehn Jahren als äußerst genau und zuverlässig erwiesen. Um den steigenden experimentellen Anforderungen und der wachsenden Datenmenge gerecht zu werden, wird die Simulationskette kontinuierlich weiterentwickelt. In den letzten Jahren haben Fortschritte im Bereich des maschinellen Lernens neue Möglichkeiten eröffnet, um die Simulations- und Analyseinfrastruktur der Hochenergiephysik weiter zu verbessern. In dieser Arbeit untersuchen wir den Einsatz moderner Algorithmen des generativen, maschinellen Lernens in verschiedenen Phasen der LHC-Simulationskette. Zunächst analysieren wir deren Potenzial zur Verbesserung der Vorwärtssimulation, insbesondere in rechenintensiven Schritten. Anschließend wenden wir generative Modelle des maschinellen Lernens auf das inverse Problem der Korrektur von Detektoreffekten an, als Alternative zu klassischen Verfahren. Wir zeigen, dass beide Aufgaben durch maschinelles Lernen mit hoher Präzision und Genauigkeit gelöst werden können und diese Ansätze damit ein grosses Potenzial zur Verbesserung zukünftiger LHC-Analysen versprechen.

Contents

Preface

The research presented in this thesis was conducted at the Institute for Theoretical Physics at Heidelberg University from November 2022 to January 2025. The contents of Chapters 3 to 5 are based on work done in collaboration with other researchers and have been previously published as

[1] Anja Butter, Nathan Huetsch, Sofia Palacios Schweitzer, Tilman Plehn, Jonas Spinner and Peter Sorrenson,
Jet diffusion versus JetGPT – Modern networks for the LHC,
SciPost Phys. Core 8, 026 (2025), arXiv:2305.10475 [hep-ph];

[2] Anja Butter, Tomáš Težo, Michael Klasen, Mathias Kuschick, Sofia Palacios Schweitzer and Tilman Plehn,
Kicking it off(-shell) with direct diffusion,
SciPost Phys. Core 7, 064 (2024), arXiv:2311.17175 [hep-ph];

[3] Luigi Favaro, Ayodele Ore, Sofia Palacios Schweitzer, Tilman Plehn,
CaloDREAM – Detector Response Emulation via Attentive flow Matching,
Scipost Phys. 18, 088 (2025), arXiv:2405.09629 [hep-ph];

[4] Anja Butter, Sascha Diefenbacher, Nathan Huetsch, Vinicius Mikuni, Benjamin Nachman, Sofia Palacios Schweitzer and Tilman Plehn,
Generative Unfolding with Distribution Mapping,
Submitted to Scipost Phys., arXiv:2411.02495 [hep-ph];

[5] Luigi Favaro, Roman Kogler, Alexander Paasch, Sofia Palacios Schweitzer, Tilman Plehn and Dennis Schwarz,
How to Unfold Top Decays,
Submitted to Scipost Phys., arXiv:2501.12363 [hep-ph].

Additionally, the author was involved in the following publication during this period,

[6] Claudius Krause (ed.), Michele Faucci Giannelli (ed.), Gregor Kasieczka (ed.), Benjamin Nachman (ed.), Dalila Salamani (ed.), David Shih (ed.) and Anna Zaborowska (ed.),
CaloChallenge 2022: A Community Challenge for Fast Calorimeter Simulation,
Submitted to Rep. Prog. Phys., arXiv:2410.21611 [physics.ins-det].

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

The Standard Model of particle physics is one of the most rigorously tested theories, describing the fundamental interactions of nature. Initially formulated in the 1960s and 70s [7–13], the theory has undergone extensive experimental validation, culminating in the discovery of the Higgs boson in 2012 [14, 15]. Despite its success in explaining a vast range of physical phenomena, the Standard Model leaves many fundamental questions unanswered, including the nature of dark matter [16, 17], the dominance of matter over antimatter in the universe [18, 19], the origins of CP-violation [20], and the existence of neutrino masses [21, 22]. These phenomena indicate the necessity of theories beyond the Standard Model that can provide a more complete picture of fundamental physics. Numerous extensions to the Standard Model have been proposed, but their validity must be tested against experimental data with extreme precision, at the five-sigma significance level.

Collider experiments provide an essential tool for testing these theories by acting as high-resolution microscopes that probe the smallest length scales of nature. The Large Hadron Collider (LHC) at CERN currently provides the highest-energy proton-proton collisions, reaching center-of-mass energies up to 13.6 TeV. However, comparing theoretical predictions to experimental measurements presents significant challenges, as the two are not directly connected through first principles and do not reside in the same phase space.

To bridge this gap, non-deterministic Monte Carlo (MC) simulations play a crucial role in propagating theoretical calculations through the full simulation chain. This chain consists of multiple steps, the hard scattering process [23, 24], parton showering [25], hadronization [25, 26], and detector effects [27, 28]. In most cases, we expect new physics signatures to show up at the hard scattering level. Each stage in the simulation chain works at a different energy scale and introduces additional degrees of freedom. Once theoretical predictions are propagated through the simulation chain, reconstruction algorithms help to map simulated and experimental data back to the hard scattering process, enabling statistical hypothesis testing to extract new or refined Standard Model parameters. At the LHC, new physics has yet to be observed.

Reliable and precise simulators are essential for precision measurements, yet they come

with a high computational cost. To minimize the impact of statistical uncertainty in MC simulations on final measurements, large-scale simulations are required. Theoretical calculations of the hard scattering process and showering are grounded in perturbative Quantum Field Theory, but these calculations are only feasible to a finite order. At some point, statistical regimes are reached where the inclusion of higher-order corrections or additional physics effects becomes relevant, further increasing computational demands.
Parallel to these challenges, robust data analysis techniques are required to efficiently process and extract insights from large datasets. Machine learning has emerged as a powerful tool in particle physics, offering both faster and more precise simulations as well as innovative analysis techniques capable of extracting the maximal amount of information from high-dimensional measurements. The use of machine learning in particle physics is not new, applications such as regressing the parton density function (NNPDF) [29] and classifier-based background suppression have been employed for years. However, recent advancements in machine learning techniques, including generative machine learning, have opened up exciting new avenues for research.
In recent years, many concepts have been introduced to utilize generative machine learning for LHC tasks, such as neural importance sampling for the hard scattering process [30–32], anomaly detection in LHC searches [33, 34] and neural simulation-based inference techniques [35–37]. We are currently transitioning from proof-of-concept to practical application, with LHC experiments beginning to incorporate generative machine learning into their analyses [38].
In addition, the interplay between physics and generative machine learning is particularly promising. Many modern machine learning architectures are inspired by physical principles, with diffusion models [39, 40] serving as a prime example. The motivation for diffusion models lies deep within the realm of non-equilibrium thermodynamics. Whereas previous precision networks like Normalizing Flows [41] rely on bijective mappings between a Gaussian distribution and a physical phase space, diffusion networks can encode such a mapping in a stochastic differential equation (SDE). This SDE can be evaluated discretely or continuously, leading to the description of Denoising Diffusion Probabilistic Models (DDPMs) [39] or Conditional Flow Matching (CFM) [40]. In this thesis, both are employed for LHC applications.
As generative machine learning continues to evolve, it is poised to revolutionize both the simulation of particle interactions and the analysis of collider data, offering new opportunities to test the Standard Model and explore physics beyond its established framework.

This thesis introduces theoretical aspects of the Standard Model, standard event generators for LHC physics and some concepts of top-quark physics in Chapter 2. A small overview of the basic definition and concepts of machine learning is reviewed in Chapter 3 with a focus on generative diffusion networks, in particular DDPMs and CFMs. Utilizing machine learning to enhance the forward simulation, i.e. augmenting simulated data, is one promising aspect of machine learning. In this thesis, machine-learning-driven fast event generation based on DDPMs and CFMs has been explored rigorously for multiple datasets, each encoding different physical phenomena. A complete presentation of the results can be found in Chapter 4. Besides data augmentation, we also studied the use of generative machine learning to enhance precision measurements. In particular, we studied the use of generative unfolding using a CFM-like setup to correct for detector effects in data in Chapter 5. Lastly, the implications of all the presented results are contextualized in a conclusion in Chapter 6 together with an outlook for future research directions.

CHAPTER **2**

High energy physics

The following chapter provides a brief overview of the physics concepts relevant to this thesis. It begins with a description of the fundamental theory of particle physics, the Standard Model, in Section 2.1, following Refs. [42,43]. This is followed by an introduction to essential concepts in collider physics, with a particular focus on a step-by-step overview of the LHC simulation chain in Section 2.2. Finally, Section 2.3 briefly reviews selected approaches to measure the top-quark mass at LHC experiments.

2.1 Standard Model

The Standard Model (SM) of particle physics provides a description of all (known) fundamental particles and their interactions in the language of a Quantum Field Theory (QFT). Any such description must respect the underlying symmetry of the Minkowski spacetime $\mathbb{R}^{3+1}$, namely the Poincare symmetry. Within the description of the SM, matter consists of spin-1/2 leptons and quarks. Interactions are mediated by gauge bosons and the full gauge group of the SM is given by

$$\mathcal{G} = SU(3)_C \times SU(2)_L \times U(1)_Y \ . \tag{2.1}$$

$SU(3)_C$ is the gauge group of the non-abelian gauge theory of Quantum Chromodynamics (QCD), the color group responsible for the strong interaction. It binds quarks into hadrons and protons and neutrons into nuclei. The Lie algebra of $SU(3)_C$ has eight generators, which define the color symmetry and satisfy specific commutation relations. The associated gauge bosons are the eight gluons, which transform in the adjoint representation of $SU(3)_C$. These gluons are massless, carry color charge, mediate the strong interaction and can couple to particles that also carry color charge, either quarks or other gluons. Leptons are colorless particles, meaning that they are $SU(3)_C$-singlets.
For a given QFT, renormalization leads to energy-dependent coupling constants. In the case of QCD the strong coupling constant α_s decreases with energy. Consequently, at high energies QCD becomes a weakly interacting theory, which we can perturbatively evaluate

in orders of α_s. At low energies however, the description of QCD is non-perturbative resulting in the necessity of finding numerical models to approximate strong interactions. In the context of collider physics the hard scattering process and parton showers are described by perturbative QCD whereas hadronization requires non-perturbative modeling.

The electroweak interaction is described by the gauge group $SU(2)_L \times U(1)_Y$. There are $3+1$ generators involved associated with 4 massless gauge bosons $W_{\mu,1/2/3}$ and B_μ. The W_μ bosons couple to left-handed fermions and right-handed anti-fermions, making the electroweak theory chiral. The coupling of the remaining B_μ boson is proportional to the particle's hypercharge. Electroweak symmetry is spontaneously broken by the Higgs mechanism

$$SU(2)_L \times U(1)_Y \to U(1)_{\mathrm{EM}} \tag{2.2}$$

when the Higgs field acquires a non-zero vacuum expectation value. Three of the four gauge bosons of the broken symmetry group acquire a mass giving rise to the massive

$$W_\mu^\pm = \frac{1}{\sqrt{2}}(W_{\mu,1} \mp iW_{\mu,2}) \qquad \text{and} \qquad Z_\mu = \cos{(\theta_W)}W_{\mu,3} - \sin{(\theta_W)}B_\mu \tag{2.3}$$

-bosons. The angle θ_W, known as the Weinberg angle, parametrizes the mixing between the gauge bosons $W_{\mu,3}$ and B_μ. It determines the relation between the weak and electromagnetic couplings. The bosons of Eq. (2.3) are the mediators of the weak interaction, which due to their masses is short ranged. The unbroken $U(1)_{\mathrm{EM}}$ group is the gauge group of Quantum Electrodynamics (QED) whose massless mediator

$$A_\mu = \sin{(\theta_W)}W_{\mu,3} + \cos{(\theta_W)}B_\mu \tag{2.4}$$

is the photon. Photons do not carry electric charge themselves, but they can couple to particles which do. Among the three forces described in this thesis, electromagnetism is the force with the longest range. In contrast to QCD, the QED coupling increases with energy, but it remains perturbative at all experimentally accessible energy scales.

The particle content of the SM, along with their transformation properties under the gauge group $\mathcal{G}$, is summarized in Tab. 2.1. The SM contains six types of quarks (up, down, charm, strange, top, and bottom) and six types of leptons (electron, electron-neutrino, muon, muon-neutrino, tau, and tau-neutrino), organized into three generations. Each quark generation contains an up-quark type and a down-quark type quark, which differ in transformation properties. Similarly, each lepton generation contains a charged lepton and a neutral neutrino. Under the electroweak gauge group $SU(2)_L \times U(1)_Y$, the left-handed components of quarks and leptons form doublets of $SU(2)_L$, whereas their right-handed counterparts transform as singlets.

Finally, we need to construct the Lagrangian of our theory. Each gauge boson contributes both a kinetic term and, in the non-abelian cases, a self-interaction term to the Lagrangian summarized in $\mathcal{L}_{\mathrm{gauge}}$. The analytical description of matter fields involve a kinetic term $\mathcal{L}_{\mathrm{kinetic,\ F}}$ including the coupling of fermions to gauge bosons as well as a Yukawa term $\mathcal{L}_{\mathrm{Yukawa}}$ describing the coupling of the Higgs bosons with right- and left-handed fermions. This coupling results in a finite fermonic mass once the gauge symmetry is spontaneously broken.
For quarks, the Yukawa couplings to the Higgs lead to a mass matrix that is not diagonal in the flavor basis. Diagonalizing the mass matrix requires independent unitary transforma-

Particle	Representation	$SU(3)_C$	$SU(2)_L$	$U(1)_Y$
Lepton $i \in \{e, \mu, \tau\}$	$(\nu_{L,i},\ e_{L,i})$	1	2	-1
	$\nu_{R,i}$	1	1	0
	$e_{R,i}$	1	1	-2
Quark $i \in \{u, d, c, s, t, b\}$	$(u_{L,i},\ d_{L,i})$	3	2	1/3
	$u_{R,i}$	3	1	4/3
	$d_{R,i}$	3	1	-2/3
Gluons	g_α	8	1	0
W's	$W_{\mu,1/2/3}$	1	3	0
B	B_μ	1	1	0
Higgs	(H^+, H^0)	1	2	1

Table 2.1: Particle content of the SM and their transformation properties under $\mathcal{G}$ adapted from Ref. [43]. The table lists the representation each particle field belongs to under each component of the gauge group before symmetry breaking. Note that $U(1)_Y$ has only one-dimensional representations. Therefore, representations under $U(1)_Y$ are labeled by their hypercharge Y. The Higgs field is a complex scalar transforming as an $SU(2)_L$ doublet with hypercharge $Y = 1/2$ consisting of an electrically charged component H^+ and a neutral component H^0. In some SM descriptions the right-handed neutrino $\nu_{R,i}$ is not included.

tions for the up-type and down-type quarks. Their mismatch leads to a non-trivial mixing in the charged-current weak interactions, described by the Cabibbo-Kobayashi-Maskawa (CKM) matrix. The CKM matrix is a unitary 3×3 matrix containing three mixing angles and one complex CP-violating phase. It governs flavor-changing transitions between quark generations and explains phenomena such as CP-violation in the weak sector. Similarly, experimental evidence from neutrino oscillations shows that neutrinos have mass and undergo flavor mixing, although this is typically not included in the SM. In analogy to the CKM matrix, neutrino mixing is described by the Pontecorvo–Maki–Nakagawa–Sakata (PMNS) matrix, which appears in the leptonic charged-current interactions. The PMNS matrix also contains three mixing angles and up to three CP-violating phases depending on the nature of neutrinos.
Moving on to the Higgs sector, there is a contribution of the kinematics and the potential of the Higgs including self-interaction $\mathcal{L}_{\text{Higgs}}$.
Combining all the mentioned aspects leads to the SM Lagrangian

$$\mathcal{L}_{\text{SM}} = \mathcal{L}_{\text{gauge}} + \mathcal{L}_{\text{kinetic,F}} + \mathcal{L}_{\text{Yukawa}} + \mathcal{L}_{\text{Higgs}} \ . \tag{2.5}$$

From the Lagrangian in Eq. (2.5), we can count the free parameters needed to have a complete description of the SM. In the gauge sector, we need to consider all coupling constants. There are in total four physical couplings, the strong coupling g_s, the electromagnetic coupling e, the weak coupling g_W and a coupling related to strong CP-violation θ.
In the fermion sector there are fermionic masses of the nine charged fermions $m_{e,i}$, $m_{u,i}$ and $m_{d,i}$ of three different generations i. Furthermore, there are four physical parameters of the CKM matrix. If neutrinos are included, we must also consider three neutrino

masses $m_{\nu,i}$ and up to six physical parameters of the PMNS matrix.
Finally, the Higgs sector contributes the Higgs mass m_H and the Higgs self-coupling λ. In total, this gives rise to 19 (+9) parameters that must be determined experimentally.

2.2 Collider Physics

There are many experiments aimed to precisely measure free parameters of the SM, probe SM predictions and search for potential Beyond Standard Model (BSM) signatures. In the context of this thesis, we focus on collider experiments with a high center of mass energy. In those high energy regimes we need to find a smart parametrization of the relativistic particles that interact in the collision.

2.2.1 Parametrization

Relativistic particles are naturally described by their four momenta in Minkowski space

$$p_\mu = (E, p_x, p_y, p_z) \qquad \text{with} \qquad g^{\mu\nu} = \mathrm{diag}(1, -1, -1, -1)\ . \tag{2.6}$$

When described in the context of LHC physics, it can be convenient to choose a particle basis that incorporates experimental properties. Defining the beam direction as z, physics should be completely symmetric around the z-axis. Therefore, it is useful to describe p_x and p_y in terms of polar coordinates

$$p_T := \sqrt{p_x^2 + p_y^2} \qquad \text{and} \qquad \phi := \arctan \frac{p_y}{p_x}\ , \tag{2.7}$$

with p_T the transverse momentum perpendicular to the z-axis and ϕ the azimuthal angle around the z-axis. Both quantities are invariant under a Lorentz boost in z-direction. As the energy E of a massive particle is not, it is often replaced by the invariant mass

$$m := \sqrt{p_\mu p^\mu} = \sqrt{E^2 - |\mathbf{p}|^2} \tag{2.8}$$

which is Lorentz invariant. Lastly, p_z is often replaced by the rapidity

$$y := \frac{1}{2} \log \left(\frac{E + p_z}{E - p_z} \right) \tag{2.9}$$

as its transformation properties under a Lorentz boost are additive and linear. However, the energy of a particle and the momentum in the beam direction are hard to measure at general-purpose colliders such as ATLAS [44] and CMS [45], whose detectors are primarily perpendicular to the beam pipe. Therefore, the pseudorapidity

$$\eta := \mathrm{arctanh} \frac{p_z}{|\mathbf{p}|} \tag{2.10}$$

is sometimes preferred over the rapidity. Although the boost-behavior of the pseudorapidity is non-linear, in the limit of massless particles

$$\lim_{m \to 0} \eta = y \tag{2.11}$$

holds. Therefore, the Lorentz transformation of the pseudorapidity is considered approximately linear. This leaves us with a complete description of a particles as

$$p_\mu = \begin{pmatrix} E \\ p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} \sqrt{m^2 + |\mathbf{p}|^2} \\ p_T \cos(\phi) \\ p_T \sin(\phi) \\ p_T \sinh(\eta) \end{pmatrix} . \tag{2.12}$$

Most results shown in this thesis make use of this parametrization and describe particles by (p_T, ϕ, η, m). As we will see later, the performance of a given machine learning task can be heavily reliant on the chosen parametrization. Sometimes we can facilitate a task by reparametrizing the given degrees of freedom.

2.2.2 Hard Scattering

At most collider experiments particle beams are accelerated to relativistic energies in a circular beam pipe until two opposite beams of potentially different particles are collided with each other. As quantum effects govern the scattering process, we can only assign a probability for two particles to interact, given that they are both within a certain area. This probability is referred to as the cross-section of a scattering process σ and has units of an area. It is connected to the event rate by

$$\frac{dN}{dt} = L\sigma \tag{2.13}$$

with L the luminosity, a measure of the density of the particle flow per unit area and time. Whereas the total cross-section is process dependent, the luminosity depends on detector geometry and experimental setup. Instead of a total event count, most analyses at collider experiments use the differential cross-section $d\sigma/dX$ in a given observable X. For example, when classically comparing the measured differential cross-section $d\sigma/dX$ to simulations, the statistical comparison can be done by a template histogram fit in bins of X.
The experiments at the LHC mostly record proton-proton collisions. Two protons are accelerated to center-of-mass energies of up to $\sqrt{s} = 13.6$ TeV. We are thus interested in $2 \to n$ scattering processes, with two initial protons with momentum p_A, p_B and n final state particles with momenta $\{p_j\}$. The differential cross-section can be calculated in QFT via

$$\begin{aligned} d\sigma = & \frac{1}{4E_A E_B |\mathbf{v}_A - \mathbf{v}_B|} \left|\mathcal{M}(p_A, p_B \to \{p_j\})\right|^2 \\ & \times \ (2\pi)^4 \delta \left(p_A + p_B + \sum_j p_j \right) \prod_j \frac{d^3 p_j}{(2\pi)^3 (2E_j)} \ , \end{aligned} \tag{2.14}$$

where E_A and E_B are the initial energies of the two protons given by $E_A = E_B = \sqrt{s}/2$ in the center-of-mass frame, $\mathbf{v_A}$ and $\mathbf{v}_B$ the respective velocities and $\mathcal{M}$ the probability amplitude or the matrix element of the process $p_A, p_B \to \{p_j\}$. The matrix element is averaged over all possible spin and color states of the initial state particles and summed over all possible spin and color states of the final state particles. The phase space integration runs over all final state particles. Momentum conservation is enforced by the δ-distribution. Complications arise from the fact that protons are not fundamental themselves but compositions of quarks and gluons. When colliding protons at high energies, the deep inelastic scattering happens between constituents of the proton of unknown momentum. A relative factor x_i relates the total available energy of one proton to the energy of the scattered parton, such that we can relate

$$E_{a/b} = x_{a/b} E_{A/B} = \frac{x_{a/b}\sqrt{s}}{2} \ . \tag{2.15}$$

Moreover, there are many possible partons a and b that could interact with each other. Therefore, the cross-section needs to be multiplied by the parton density functions (PDF) $f_i(x_i)$, the probability of finding parton i with momentum fraction x_i for $i \in \{a, b\}$. This would then correspond to the cross-section of parton a and b to interact. In practice however we need to sum over all possible partons a and b and their corresponding partonic cross-sections. We end up with

$$\sigma = \sum_{a,b} \int_0^1 dx_a \int_0^1 dx_b f_a(x_a) f_b(x_b) \sigma(p_a, p_b \to \{p_j\}) \ . \tag{2.16}$$

The PDFs cannot be determined from first principles. However, their scale dependence is governed by the Dokshitzer–Gribov–Lipatov–Altarelli–Parisi (DGLAP) equations, which describe that PDFs evolve with the transferred momentum Q^2 as

$$\frac{df_a(x, Q^2)}{d \log Q^2} = \sum_b \int_x^1 \frac{dz}{z} \frac{\alpha_s(Q^2)}{2\pi} f_b(\frac{x}{z}, Q^2) P_{a\leftarrow b}(z) \ . \tag{2.17}$$

Here, z is the momentum fraction retained by the daughter parton a after the splitting of parton b, and $P_{a\leftarrow b}(z)$ are the universal splitting kernels. They describe the probability that a parton of type b splits into a parton of type a, such that a carries a fraction z of the momentum of the original parton b. In leading-order approximation, the splitting kernels depend only on the momentum fraction z and the parton types a and b. A more detailed description of parton evolution can be found in e.g Ref. [46].
In practice, PDFs are extracted from experimental data as functions of the parton's momentum fraction x and the hard scale Q^2 and in a second step fitted to the DGLAP equations [29].
Finally, to simulate hard scattering processes, events must be sampled from Eq. (2.16). Among popular event generators for that task are MadGraph [23], Sherpa [24] and Pythia8 [25].

2.2.3 Parton Shower

While the hard scattering defines the primary interaction, it does not account for additional QCD radiation that occurs as partons evolve. Both the incoming and outgoing partons can radiate gluons or, in some cases, quark-antiquark pairs, leading to a cascade known as a parton shower. Soft emissions occur when the radiated gluon carries very little energy, while collinear emissions happen when the emitted parton is nearly parallel to the parent parton. The probability of such emissions is enhanced due to divergences in QCD. Since the angular and energy resolution for hadronic final states at LHC experiments are typically of order $\mathcal{O}(0.1)$ in both angle and energy energy [44,45], we cannot experimentally distinguish between an event where a parton radiates a soft or collinear gluon and one where it does not. Moreover, from a theoretical perspective, infrared-safe observables are explicitly constructed to be insensitive to such emissions [46]. Therefore, we must

include these effects in our event generation to obtain realistic predictions.
In the soft and collinear limits, the squared QCD matrix element for an $(n+1)$-parton final state factorizes into the matrix element for a n-parton configuration, convoluted with the appropriate splitting kernel [46] introduced in Eq. (2.17). In particular, the cross-sections are related by

$$\sigma_{n+1} \approx \sigma_n \int dz \frac{dQ^2}{Q^2} \frac{\alpha_s(Q^2)}{2\pi} P_{a\leftarrow b}(z) \; . \tag{2.18}$$

This universal behavior allows the radiation pattern to be described probabilistically in terms of successive splittings, each governed by the splitting kernel and the running of the strong coupling α_s. The expression on the right-hand side can be interpreted as a differential probability for a parton of type b to split into a parton a

$$d\mathcal{P}_{b\to a} = \frac{dQ^2}{Q^2} \frac{\alpha_s(Q^2)}{2\pi} P_{a\leftarrow b}(z) dz \; . \tag{2.19}$$

We distinguish between four different splitting kernels. A quark radiating a gluon $q \to qg$ is described by $P_{q\leftarrow q}$ and $P_{g\leftarrow q}$, a gluon splitting into an quark-antiquark pair $g \to qq$ is described by $P_{q\leftarrow g}$ and a gluon radiating a gluon $g \to gg$ is described by $P_{g\leftarrow g}$. The concrete calculation of the splitting kernels is described in great detail in e.g. Ref. [46].
Parton showers are not limited to the outgoing partons. In hadronic collisions, the incoming partons can also radiate before the hard scattering takes place, namely initial-state radiation (ISR). Shower algorithms handle this by interpreting the splitting as evolving backwards in time. This backward evolution makes use of the same splitting probabilities of Eq. (2.19), but applied in reverse. To reconstruct a parton b that entered the hard process, we consider the possibility that it originated from a splitting $a \to b + X$ at a higher scale. This backward evolution is constrained by the parton distribution functions, which encode the probability of finding a parton a inside the proton with a given momentum fraction such that the overall probability becomes

$$d\mathcal{P}_{a\to b} = \frac{dQ^2}{Q^2} \frac{\alpha_s(Q^2)}{2\pi} P_{a\leftarrow b}(z) \frac{f_a(x/z, Q^2)}{f_b(x, Q^2)} dz \; . \tag{2.20}$$

Modern event generators, such as Pythia8, implement parton showers using this probabilistic picture. Starting from the hard scattering process, a parton b is allowed to split according to the so called Sudakov factor

$$\Delta_b(Q, Q_0) = \exp\left(-\int_Q^{Q_0} \frac{d\tilde{Q}^2}{\tilde{Q}^2} \sum_a \int dz \frac{\alpha_s(\tilde{Q}^2)}{2\pi} P_{a\leftarrow b}(z) \right) \; , \tag{2.21}$$

which encodes the probability that a parton evolves from a high scale Q_0 down to a lower scale Q without radiating a parton. In Pythia8, Sudakov factors are sampled uniformly between 0 and 1 and Eq. (2.21) is numerically inverted to solve for the energy scale Q at which the next emission occurs. Each parton in the event undergoes this procedure iteratively, splitting or not based on the Sudakov factor, until the evolution reaches a cutoff scale below which non-perturbative physics takes over.

2.2.4 Hadronization

Once the momentum transfer of the parton splitting is of order $\mathcal{O}(1\ \text{GeV}^2)$, non-perturbative QCD regimes are reached and partons start to hadronize. Whereas the

hard scattering process and the parton shower can be described from first principles, hadronization cannot. Instead, phenomenological models are defined to numerically provide a description of the hadronization process. There are two models typically used for LHC tasks.
The Lund String Model [47] describes hadronization as a process where color-connected quarks and gluons form a string-like structure due to the strong force. As the string stretches or as the distance r between the string ends increases, it stores potential energy

$$V(r) = \kappa r \ , \tag{2.22}$$

where $\kappa \approx 1$ GeV/fm is the string tension, a phenomenological constant. Eventually, the potential causes the string to break into smaller pieces, as it becomes energetically favorable to form a new quark-antiquark pair. Thus, hadronization is modeled as repeated string breaking, where each break creates a new quark–antiquark pair that forms a hadron with a neighboring parton. The model specifies how the energy and momentum are assigned to these hadrons. This process is governed by a fragmentation function, the probability for a hadron to take a fraction z of the remaining string momentum given the hadron's flavor and its transverse momentum p_T sampled from a Gaussian distribution with tunable width. The Lund fragmentation function is given by

$$f(z) \propto \frac{(1-z)^A}{z} \exp\left(-\frac{Bm_T^2}{z}\right) \ , \tag{2.23}$$

with m_T the transverse mass of the hadron and A and B tuneable model parameters. Once the remaining energy in the string is too small to produce additional hadrons, the string forms the final hadrons if momentum and quantum numbers can be conserved. If not, the chain of string breaks is rejected and the procedure is repeated. The Lund string model is widely used in event generators like Pythia8 [25].

Cluster Fragmentation is an alternative hadronization model where color-neutral clusters are formed from the products of parton showers. These clusters, typically low-mass gluon-gluon or quark-antiquark pairs, decay into hadrons based on phase space and mass thresholds. It is commonly used in event generators like Herwig [26]. Both models rely heavily on fine-tuning, meaning numerous model dependent parameters that can be adapted or tuned to match measured data best.
Some hadrons have a very short lifetime and decay before reaching the detector. These decays must be modeled in simulations to determine the final set of stable particles that can interact with the detector material. Final-state hadrons are considered stable within the detector because they typically decay inside its volume or live long enough to be detected.

2.2.5 Detector

Simulating the detector response of a given process comes with the highest computational cost [48], making it a severe bottleneck for the number of simulations available in high energy physics (HEP) experiments. General-purpose detectors like ATLAS and CMS are composed of multiple layers, each designed for specific particle detection. Closest to the interaction point is the tracker, which detects charged particles by measuring their trajectories. Surrounding it is the electromagnetic calorimeter (ECAL), which primarily measures energy from photons and electrons, followed by the hadronic calorimeter

(HCAL), which captures energy from hadrons. The outermost layer consists of muon chambers that detect muons, which can penetrate the inner detector layers. A widely used toolkit for simulating detector responses is Geant4 [27]. It takes the output from event generators like Pythia8 and propagates them through the detector, modeling their interactions with the detector material at every stage of their passage. Geant4 provides detailed tracking through complex detector geometries, accounting for effects such as energy loss, multiple scattering, and secondary particle production. It accurately simulates electromagnetic and hadronic interactions, including ionization, bremsstrahlung, hadron showers, and decays. Finally, Geant4 converts simulated energy deposits into signals, mimicking real detector readouts, such as hits in calorimeters or ionization in tracking detectors.
For many analyses detailed detector simulations are not crucial such that fast detector simulators can be a cheap alternative. One of which is Delphes [28]. Instead of detailed Geant4-style tracking, it simply applies parametrized resolutions, efficiencies, and smearing functions to the outputs from event generators like Pythia8.

2.2.6 Reconstruction

After propagating a given theory through the entire simulation chain, we are finally at a point where simulations live in the same physical phase space as data. To translate detector readouts back to physical particles, reconstruction algorithms are used. These algorithms vary between experiments, but all of them reconstruct particle-level objects by combining information from different detector components. They identify charged particles using the tracker, neutral hadrons from the hadronic calorimeter, and photons from the electromagnetic calorimeter. Delphes [28] for example comes with its own reconstruction algorithm.
Most physics analyses at collider experiments are studying the hard scattering process. Thus, once all final state particles are reconstructed, we further try to reconstruct the entire decay chain by enforcing signal-dependent selection criteria. A complication arises from the fact that partons produced at the hard scattering process likely shower and hadronize into $\mathcal{O}(10-100)$ final-state particles depending on its initial energy. To overcome this complication, there exists various algorithms that cluster detected particles into so-called jets. Usually, they cluster two objects (particles or subjets) i and j according to a distance measure in momentum space defined as

$$d_{ij} = \frac{\Delta R_{ij}^2}{R^2} \min({p_{T,i}}^{2n}, {p_{T,j}}^{2n}) \qquad \text{with} \qquad \Delta R_{ij} = \sqrt{(\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2} \;, \tag{2.24}$$

where the jet radius R is a fixed parameter inputted to the jet algorithm. If d_{ij} is smaller than the distance of object i to the beam pipe $d_{iB} = {p_{T,i}}^{2n}$, both objects are clustered to form a subjet. This process is repeated until all particles are clustered or a fixed number of jets are formed. There are three commonly used clustering algorithms kt (n=1) [49], anti-kt (n=-1) [50] and the Cambridge Aachen algorithm (n=0) [51]. Depending on the algorithm, jets have slightly different properties.
An alternative description of jets is offered by Xcone [52]. Unlike traditional jet algorithms, which use a fixed cone or distance measure, the Xcone algorithm adapts the size of the jet cone based on the local energy flow in the event. In the Xcone approach, the cone size around each particle can dynamically change to ensure that the jet is energy-conserving and robust against multiple jets overlapping. The algorithm starts by defining an initial cone around each particle and then iteratively adjusts the cone size to group energy

deposits into a well-defined jet.
A toolkit that offers all standard jet algorithms is fastjet [53], which is also interfaced with Delphes [28].

2.2.7 Statistical testing

Finally, we would like to connect a experimental measurement with an analytical description of the underlying theory. However, direct analytical expressions for these predictions become intractable due to the probabilistic nature of the MC and numerical assumptions. Therefore, we test whether the simulated theory predictions accurately describe the observed data by e.g. comparing binned, one-dimensional distributions using statistical fits. These fits incorporate both statistical and systematic uncertainties, which are combined as

$$\sigma = \sqrt{\sigma_{\text{stat}}^2 + \sigma_{\text{sys}}^2}\,. \tag{2.25}$$

If the MC simulation deviates from the observed data by more than 5σ and the accuracy of the theoretical prescription is verified, a discovery is claimed, namely that the SM alone cannot describe the observation. To test new physics models, simulations incorporating BSM parameters must be generated and passed through the full simulation chain. A new hypothesis test is then conducted to determine whether the BSM model provides a better fit to the data.

2.3 Measuring the top-quark mass

The LHC is a top-quark factory as it is expected to produce billions of top-quarks in its runtime of several decades [54]. The top-quark is the heaviest known fundamental particle, with a mass of approximately 172.5 GeV [55]. Due to its large mass, it has the strongest coupling to the Higgs boson, making it an ideal candidate to test electroweak symmetry breaking. In many LHC analyses, the top-quark serves either as a leading background that needs to be accounted for or as the primary object of study. Unlike lighter quarks, which undergo parton showering and hadronization, the top-quark decays almost instantly. A concrete understanding of top-quark physics is crucial for testing the Standard Model. Several key parameters of the top-quark need to be determined precisely, one of the most important being its mass. At the LHC, various experimental methods exist to measure the top-quark mass. A common approach involves studying top-antitop ($t\bar{t}$) decays. At the LHC, the leading production channel of $t\bar{t}$-pairs is gluon fusion, $gg \to t\bar{t}$, due to high gluon density in high energetic protons. The subleading production mechanism is quark-antiquark annihilation, $q\bar{q} \to t\bar{t}$.
Regarding the decay, one side of the decay is typically assumed to be leptonic ($t \to b\ell\nu_\ell$), where the emitted lepton acts as a tag for one top-quark decay, indicating the presence of the other top-quark. Once the $t\bar{t}$-decay is identified, the second top-quark can decay leptonically ($t \to b\ell\nu_\ell$) or hadronically ($t \to bq\bar{q}'$). All hadronic decay products shower and hadronize, so that the associated final state objects are jets. Each decay channel of the second top-quarks comes with its own challenges. Whereas in fully leptonic $t\bar{t}$-decays kinematic information is lost due to at least two neutrinos, which cannot be measured, semileptonic $t\bar{t}$-decays come with 4 jets whose kinematic properties are harder to reconstruct precisely. Starting with fully leptonic $t\bar{t}$-decays, a common experimental

approach [56] for a top-quark measurement is to reconstruct the invariant mass of the b-quark-lepton system

$$m_{\ell b}^2 = (p_\ell + p_b)^2 = (E_\ell + E_b)^2 - (\vec{p_\ell} + \vec{p_b})^2 \ , \tag{2.26}$$

originating from the same mother top-quark. Note that the full top-quark cannot be reconstructed in this channel as the accompanying neutrino is undetected.
The b-quark energy in the top-quark rest frame is given by

$$E_b = \frac{m_t^2 + m_b^2 - m_W^2}{2m_t} \approx \frac{m_t^2 - m_W^2}{2m_t} \tag{2.27}$$

due to four-momentum conversation. Often, the b-quark mass is neglected. Similarly, we can calculate the lepton energy in the W-boson rest frame as

$$E_l^* = \frac{m_W^2 + m_\ell^2}{2m_W} \approx \frac{m_W}{2} \ . \tag{2.28}$$

The invariant mass $m_{\ell b}$ is maximal when the lepton and the b-quark are emitted back-to-back in the top-quark rest frame, which gives the precise endpoint of the kinematic distribution

$$m_{\ell b,\max} = \sqrt{\frac{(m_t^2 - m_W^2) m_W^2}{2m_t}} \ . \tag{2.29}$$

Whereas in reality detector resolution, background contamination and finite particle decay widths cause migration over the kinematic edge, in leading-order this is a theoretical upper bound. Higher-order terms as well as off-shell effects can smear out the kinematic edge and, hence, need to be included in the MC simulation of the hard scattering for a valid comparison to data. Those aspects are discussed in more detail in Section 4.2.

However, beyond inaccuracies in simulation, there is also a conceptual issue when measuring the top-quark mass. In a QFT such as the SM, parameters like mass are not physical observables by themselves. Due to ultraviolet (UV) divergences in loop diagrams, the mass must be renormalized, meaning it is defined within a particular scheme that consistently absorbs these divergences [57]. As a result, there exist multiple valid definitions of the top-quark mass, depending on how the renormalization is carried out.
Thus, we need to understand how to interpret a measurement as described above, in which a template fit is performed between data and MC simulations to extract the MC top-quark mass $m_{t,\mathrm{MC}}$ that best describes the observed $m_{\ell b}$. In MC simulations, the top-quark mass parameter determines the value of the mass that enters the top-quark propagator in the hard scattering process, before the top-quark decays. Since observables such as $m_{\ell b}$ are sensitive to effects from parton showers and hadronization, the extracted MC mass is implicitly influenced by these components and therefore does not correspond to a renormalized field-theoretic mass [57]. To highlight the difference from well-defined renormalized masses, it is important to note that parton showers, as implemented in Monte Carlo generators, simulate real emissions of soft and collinear radiation but do not include virtual corrections such as self-energy loops. As a result, the top-quark propagator used in the simulation is not corrected by field-theoretic self-energy contributions, and the mass parameter remains unchanged throughout the showering process. Thus, the MC mass does not intrinsically incorporate radiative corrections.

Only after tuning parton shower and hadronization parameters of the MC simulation to reference processes in experimental data, are the effects of radiative and non-perturbative corrections effectively absorbed into the definition of $m_{t,\mathrm{MC}}$ [57]. Therefore, the MC mass should be interpreted as an effective parameter that encodes both the leading-order hard process and model-dependent QCD effects. Any translation of $m_{t,\mathrm{MC}}$ into a theoretically well-defined mass scheme, have shown to introduce an irreducible uncertainty of approximately $(0.5 - 1.0)$ GeV [58].

There are alternative ways to measure a more meaningful top mass in semileptonic $t\bar{t}$-decays as presented in Ref. [59]. In the boosted regime, top-quark decay products are highly collimated and can be reconstructed as a single fat jet. It has been shown, that the peak position of the invariant mass of this jet can be related to a field-theoretically well-defined mass parameter. Despite the advantages of boosted topologies, both experimental and theoretical challenges remain in performing such a measurement. In particular, theoretical predictions are made at the particle level and assume idealized conditions. To enable a meaningful comparison, detector effects must be corrected for by unfolding the measured jet mass distribution. An issue we tackle in Section 5.1.

CHAPTER **3**

Machine Learning

This chapter provides a description of the machine learning (ML) concepts and tools employed in this thesis. Starting with basics definitions in Section 3.1, we briefly explore classification as a common ML task in Section 3.2 followed by a short introduction into uncertainty-aware Bayesian neural networks in Section 3.3. We proceed with an in-depth discussion of generative diffusion models in Section 3.4 and more general diffusive distribution mapping techniques in Section 3.5.

3.1 The basics

Machine learning can be described as the process of fitting mathematical models to high-dimensional data. At the core of these models are often deep neural networks, structured as a sequence of linear transformations (matrix multiplications) combined with nonlinearities (activation functions).

Neural Networks

Given an m-dimensional point sampled from a physical phase space $x \sim p(x)$ as input to a deep neural network, we define a hidden layer as

$$y_1(x) = \text{activation}(W_1 x + b_1) \;, \tag{3.1}$$

with a hidden dimension of n, where the weight matrix W_1 is an $m \times n$ matrix of learnable parameters, and b_1 is an n-dimensional learnable bias vector. Furthermore, we define the number of hidden layers N as the number of affine transformations applied to x. Nonlinearities are introduced after each linear mapping, except for the last layer, which may or may not include one depending on the application. Typical activation functions include the sigmoid function and variations of the ReLU function, among others. Activation functions are crucial in deep learning, as without them, the network could only model linear relationships. However, a complete overview of activation functions lies outside the scope of this thesis. Applying the full stack of N layers and $N-1$ activation functions to x results in the network output:

$$z = W_N y_{N-1}(x) + b_N \ , \tag{3.2}$$

where y_{N-1} is the output of the previous transformations. Input and output dimensions of the neural network are given by $\dim(x)$ and $\dim(z)$ respectively.

Training

Usually, the entries of W and b are initialized with random noise. In order for them to capture patterns from the data, we need to train the neural network. The training objective, or loss, is a scalar function of z. Its concrete form is problem-dependent and is denoted by $\mathcal{L}$. We optimize the loss $\mathcal{L}$ with respect to the learnable parameters in W and b using gradient descent. Gradient descent is an iterative optimization algorithm used to minimize a loss function by updating the model parameters in the direction of the steepest descent. For each parameter θ, the update rule for gradient descent is given by

$$\theta_{i+1} = \theta_i - \lambda \nabla_\theta \mathcal{L}(\theta) \ , \tag{3.3}$$

where $\lambda > 0$ is the learning rate, which controls the step size of the update, and $\nabla_\theta \mathcal{L}(\theta)$ is the gradient of the loss function with respect to the trainable parameters θ. The gradient is computed via backpropagation. In standard gradient descent, the gradient $\nabla_\theta \mathcal{L}(\theta)$ is computed over the entire dataset, which can be computationally expensive. Stochastic Gradient Descent (SGD) approximates this by computing the gradient using only a small, randomly selected subset (batch) of the data. The update rule for SGD is given by

$$\theta_{i+1} = \theta_i - \lambda \nabla_\theta \mathcal{L}_{\text{batch}}(\theta) \ , \tag{3.4}$$

where $\mathcal{L}_{\text{batch}}$ is the loss computed on a single data sample or batch. While SGD introduces statistical noise into the optimization process, it often leads to faster convergence and better generalization by enabling the model to escape sharp local minima. An extension of vanilla SGD is Adam [60], an adaptive optimization algorithm that improves stability and efficiency.
The training procedure is repeated for a fixed number of iterations or until the validation loss, computed on a statistically independent subset of the training data not used for updating the model weights, converges. Finally, the performance of the neural network is evaluated on an independent test dataset.
Typical training hyperparameters that are tuned include the number of iterations (epochs), batch size, and learning rate. Additionally, a learning rate scheduler is often employed to dynamically adjust the learning rate during training based on the current iteration number or validation loss. To mitigate overfitting, i.e. preventing the neural network from merely memorizing the training data, we can introduce dropout [61], a regularization technique where random entries in the network are set to zero during training with a predefined Bernoulli probability p_{drop}. In addition, there are also regularization techniques that discourage large weights in a neural network by adding a penalty term, proportional to either the squared magnitude or the absolute value of the weights, to the loss function. This approach is known as weight decay and is governed by a hyperparameter α, which controls the relative contribution of the penalty term to the overall loss.

3.1.1 Special Network Architectures

So far, we have considered the simple case where the neural network architecture is fully connected or dense. This means that every component of an input or hidden vector in a given layer is connected to every component of the subsequent layer through weight matrices. However, adaptations of this basic architecture are often introduced to improve performance in specific tasks or to intrinsically biasing the network to incorporate known physical symmetries. For example, in some architectures, certain connections are explicitly set to zero, ensuring that components of hidden vectors only influence their neighboring components in the subsequent layer. In addition, the definition of the network output can vary. For instance, we can interpret the network output $z(x)$ as a correction to the input x, provided they share the same dimensionality. In this case, we define our output as

$$\tilde{x} = x + z(x) \; . \tag{3.5}$$

This is known as a residual connection or skip connection, as the input x not only serves as an input to the network but also bypasses intermediate transformations and directly influences the output. While such connections can theoretically be applied at various points in a network architecture, in the context of this work, we only consider the specific form given in Eq. (3.5). Beyond architectural modifications, specialized layers are often introduced to enhance performance. One such technique is normalization, which stabilizes training and improves generalization. One commonly used normalization methods is batch normalization (BatchNorm) [62], which normalizes each feature across a batch, ensuring that activations remain within a stable range. Another method is layer normalization (LayerNorm) [63], which normalizes each individual sample across its input features.

Transformers

The introduction of transformer layers [64] has revolutionized the field of natural language processing and they also have been shown to enhance the performance in various HEP applications [1, 3, 65]. They operate on sequences and rely on attention mechanisms to assign importance to relationships between pairs of elements within a sequence. In the context of HEP, we may consider a batch containing samples of $N_\text{particles}$ particles, each represented by their four-momentum components. This results in an input shape of $(N_\text{batch}, N_\text{particles}, 4)$. Usually, we would start by mapping the input to a higher dimensional embedding with dimensionality l. To compute what we call self-attention, we first define three matrices, queries Q, keys K, and values V. Given an input vector x, they are defined as

$$Q(x) = W_Q x \qquad K(x) = W_K x \qquad V(x) = W_V x, \tag{3.6}$$

where W_Q, W_K, W_V are weight matrices with learnable parameters. If each input vector has a dimensionality of l, these matrices typically have dimensions $k \times l$, where k is a hyperparameter. The self-attention matrix is then computed by

$$A(x) = \text{softmax}\left(\frac{Q(x)K(x)^T}{\sqrt{k}}\right) , \tag{3.7}$$

which results in a matrix of shape $(N_\text{particles}, N_\text{particles})$. The softmax is applied along the rows of the matrix product. Each row in $A(x)$ represents the attention weights for a given particle relative to all other particles. These weights are then applied to the values

$$S(x) = A(x)V(x), \tag{3.8}$$

resulting in an output tensor of shape $(N_\text{particles}, k)$. This output is then projected back to the output space of desired dimensionality using a learnable linear transformation. In practice, multiple attention heads can be used, each computing separate attention scores and outputs. These are concatenated and transformed using another linear layer, allowing the network to learn different aspects of the correlations among particles. In scenarios where additional information c is available, cross-attention is employed. Instead of computing self-attention within a single set of inputs, attention is computed between two different inputs

$$A(x,c) = \text{softmax}\left(\frac{Q(x)K(c)^T}{\sqrt{k}}\right) , \tag{3.9}$$

namely between the condition c and the input x. The cross-attended output is then given by

$$S(x,c) = A(x,c)V(c), \tag{3.10}$$

where the values now come from the condition c, allowing the network to integrate additional information into the particle representation. Since transformers are intrinsically permutation invariant and a particular physics problem might not, it can be useful to add positional information into the embedding, s.t. this symmetry is explicitly broken.

3.2 Classification

A popular machine learning task is classification, in particular binary classification. In the case of binary classification there are two distinct classes labeled by $y = 0$ or $y = 1$. In all applications considered in this thesis, we set the priors $p(y = 0) = p(y = 1) = 0.5$. The goal of a classifier is to assign a probability that a high-dimensional input vector x belongs to class 1. This means we would like the classifier output to approximate

$$C_\theta(x) \approx p(y = 1|x) \qquad \text{and consequently} \qquad 1 - C_\theta(x) \approx p(y = 0|x) \tag{3.11}$$

with learnable parameters θ. To ensure that the classifier output behaves as a probability, a sigmoid function can be applied to the output of the last layer. The sigmoid function forces the classifier output to lay between $0 < C_\theta(x) < 1$. The probability for a given x to be of class y is hence given by

$$p(y|x,\theta) = C_\theta(x)^y (1 - C_\theta(x))^{1-y} , \tag{3.12}$$

which we refer to as the likelihood. We would like to learn θ, s.t. the likelihood for all $(x, y) \sim p_\text{train}(x, y)$ is maximal. This is equivalent to minimizing the negative log-likelihood

$$\mathcal{L}_\text{class} = -\Big\langle \log p(y|x,\theta) \Big\rangle_{(x,y)\sim p_\text{train}(x,y)}$$

$$= -\Big\langle y \log(C_\theta(x)) + (1-y)\log(1-C_\theta(x)) \Big\rangle_{(x,y)\sim p_\text{train}(x,y)} \, . \tag{3.13}$$

which is the typical loss function used to train a binary classifier. It is also called the binary cross entropy loss. The classifer is converged when

$$\begin{aligned} \frac{\delta \mathcal{L}}{\delta C_\theta} &= -\frac{p(x|y=1)}{C_\theta(x)} - \frac{p(x|y=0)}{1-C_\theta(x)} \overset{!}{=} 0 \\ \to C_\theta(x) &= \frac{p(x|y=1)}{p(x|y=0)+p(x|y=1)} \end{aligned} \tag{3.14}$$

This implies that we can relate the output of a converged classifier to the likelihood ratio

$$w(x) := \frac{p(x|y=1)}{p(x|y=0)} = \frac{C_\theta(x)}{1-C_\theta(x)} \, . \tag{3.15}$$

In the context of this work we exploit this property to reweight samples of a distribution $x \sim p_0(x)$, s.t. they follow a different distribution $p_1(x)$. This can be achieved by training a classifier between samples of the $p_0(x)$ and $p_1(x)$ and weighting $x \sim p_0(x)$ by Eq. (3.15). It is important to note that the analytical form of $p_0(x)$ and $p_1(x)$ does not have to be known as long as we have samples of both distributions. Furthermore, Eq. (3.15) implies that the weights diverge for $p(x|y=0) \to 0$. Logically, this makes sense as we cannot reweight empty phase space regions. Therefore, we need to ensure that the support of $p(x|y=1)$ is a subset of the support of $p(x|y=0)$.

3.3 Bayesian Neural Networks

The content of this section was finalized in collaboration with Anja Butter, Nathan Huetsch, Tilman Plehn, Jonas Spinner and Peter Sorrenson. It is adapted from Ref. [1].

Any neural network task comes with associated uncertainties, for instance from a limited amount of training data, a lack of model flexibility, or even training data which we know cannot be trusted. This means, that there should be an associated uncertainty to the extracted output, ideally in form of a second map over the target phase space. This problem has been tackled for the task of density estimation with bijective normalizing flows through a Bayesian network extension [66], which can be combined with other measures, like conditional training on augmented data [41].

The idea behind Bayesian neural networks (BNNs) is to train network weights as distributions and evaluate the network by sampling over these distributions. This will provide a central value and an uncertainty for the numerically defined network output [67–69]. Because general Markov-Chain-Monte-Carlo(MCMC)-methods are expensive for large networks, we use variational inference [70] to learn Gaussian approximations for each weight distribution. Because of the non-linear nature of the network this does not mean that the network output has to come with a Gaussian uncertainty [71].

We repeat the main steps in deriving the Bayesian loss for any neural network approximating, for instance, a density map $\rho(x) \approx \rho_\theta(x)$ following Ref. [72]. The expectation value is defined as

$$\langle\, \rho \,\rangle(x) \equiv \langle\, \rho \,\rangle = \int d\rho \; \rho \; p(\rho) \qquad \text{with} \qquad p(\rho) = \int d\theta \; p(\rho|\theta) \; p(\theta|x_\text{train}) \; , \tag{3.16}$$

where we omit the x-dependence. We use the variational approximation to approximate

$$p(\rho) = \int d\theta\; p(\rho|\theta)\; p(\theta|x_\text{train}) \approx \int d\theta\; p(\rho|\theta)\; q(\theta)\;, \tag{3.17}$$

where $q(\theta)$ is also a function of x. The variational approximation step requires us to minimize

$$\begin{aligned} \mathcal{L}_\text{BNN} = \text{KL}[q(\theta), p(\theta|x_\text{train})] &= \left\langle \log \frac{q(\theta)}{p(\theta|x_\text{train})} \right\rangle_q \\ &= \int d\theta\; q(\theta)\; \log \frac{q(\theta)}{p(\theta|x_\text{train})} \\ &= -\int d\theta\; q(\theta)\; \log p(x_\text{train}|\theta) + \text{KL}[q(\theta), p(\theta)] + \text{const}\;, \end{aligned} \tag{3.18}$$

where we use Bayes' theorem to transform the intractable $p(\theta|x_\text{train})$, introducing the prior $p(\theta)$ for the network weights. This so-called Evidence-Lower-Bound (ELBO) loss combines a likelihood loss with a regularization term, their relative size fixed by Bayes' theorem.

It turns out that for sufficiently deep networks we can choose $q(\theta)$ as uncorrelated Gaussians per network weight [69], such that the training parameters are a set of means and standard deviations for each network weight. Compared to the deterministic network, its Bayesian version is twice the size, but automatically regularized, keeping the additional numerical effort minimal. While $p(\theta)$, also chosen as a Gaussian, is formally defined as a prior, we emphasize that in our case the step from the prior to the posterior has nothing to do with Bayesian inference. The Gaussian width of $p(\theta)$ can be treated as a network hyperparameter and varied to improve the numerical performance. We typically find that the result is stable under varying the width by several orders of magnitude, and a width of one works well.

Switching a deterministic network into its Bayesian version includes two steps, (i) swap the deterministic layers to the corresponding Bayesian layers, and (ii) add the regularization term to the loss. For the latter, one complication arises. We estimate the complete loss from a dataset including N events in M batches, which means the likelihood term is summed and then normalized over M batches, while the regularization term comes with the complete prefactor $1/N$.

To evaluate the Bayesian network we need to again sample over the network weight distribution. This way we guarantee that the uncertainty of the network output can have any functional form. The number of samplings for the network evaluations can be chosen according to the problem. Unless stated otherwise, we choose 30 for all problems discussed in this work. To compare the Bayesian network output with a deterministic network output we can either go into the limit $q(\theta) \to \delta(\theta - \theta_0)$ or only evaluate the means of the network weight distributions.

3.4 Generative Diffusion Models

The content of this section was finalized in collaboration with Anja Butter, Nathan Huetsch, Tilman Plehn, Jonas Spinner and Peter Sorrenson. It is adapted from Ref. [1].

Typical generative models map simple latent distributions to a phase space distribution encoded in the training data,

$$r \sim p_{\text{latent}}(r) \quad \longleftrightarrow \quad x \sim p_\theta(x|\theta) \approx p_{\text{data}}(x) \; . \tag{3.19}$$

The last step represents the network training, for instance in terms of a variational approximation of $p_{\text{data}}(x)$. The latent distribution is typically a standard multi-dimensional Gaussian,

$$p_{\text{latent}}(r) = \mathcal{N}(r; 0, 1) \; . \tag{3.20}$$

We focus on the case where the dimensionalities of the latent space r and the phase space x are identical, and there is no lower-dimensional latent representation. For these kinds of dimensionalities, bijective network architectures are promising candidates to encode precision correlations. For strictly symmetric bijective networks like Invertible Neural Networks (INNs) the forward and backward directions are inverse to each other, and the network training and evaluation is symmetric. In the context of this thesis INNs are frequently used as benchmarks, but they are not studied in greater detail. Instead, we introduce two diffusion models in Section 3.4.1 and Section 3.4.2 which follow the basic structure of Eq. (3.19) by casting the mapping as a time-dependent transformation, one with a discrete and one with a continuous time evolution. Because many applications require full control and a conservative and reliable uncertainty estimation of neural networks, we develop Bayesian versions of the two generative models.

3.4.1 Denoising Diffusion Probabilistic Model

Architecture

Denoising Diffusion Probabilistic Models (DDPM) [39] transform a model density by gradually adding Gaussian noise. This setup guarantees that the network links a non-trivial physics distribution to a Gaussian noise distribution, as illustrated in Eq. (3.19). The task of the reverse, generative process is to denoise this diffused data. The structure of diffusion models considers the transformation in Eq. (3.21) a time-dependent process with $t = 0 \ldots T$,

$$p_\theta(x_0|\theta) \quad \underset{\leftarrow\text{backward}}{\overset{\text{forward}\rightarrow}{\longleftrightarrow}} \quad p_{\text{latent}}(x_T) \; . \tag{3.21}$$

The DDPM discretizes the time series in Eq. (3.21) in the forward direction and encodes it into a neural network for the backward direction. We start with the forward process, which turns the physical distribution into noise. The corresponding joint distribution is factorized into discrete steps,

$$\begin{aligned} p(x_1, ..., x_T|x_0) &= \prod_{t=1}^{T} p(x_t|x_{t-1}) \\ \text{with} \qquad p(x_t|x_{t-1}) &= \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t) \; . \end{aligned} \tag{3.22}$$

Each conditional step $p(x_t|x_{t-1})$ adds noise with variance β_t around the mean $\sqrt{1-\beta_t}x_{t-1}$. The combination of x_t as a variable and the mean proportional to x_{t-1} implies that the successive steps can be combined as Gaussian convolutions and give the closed form

$$\begin{aligned} p(x_t|x_0) &= \int \prod_{i=1}^{t-1} dx_i \; p(x_t|x_{t-1}) p(x_i|x_{i-1}) \\ &= \mathcal{N}(x_t; \sqrt{1-\bar{\beta}_t} x_0, \bar{\beta}_t) \qquad \text{with} \qquad 1 - \bar{\beta}_t = \prod_{i=1}^{t} (1-\beta_i) \; . \end{aligned} \tag{3.23}$$

The scaling of the mean with $\sqrt{1-\beta_t}$ prevents the usual addition of the variances and instead stabilizes the evolution of the Gaussian over the time series. The variance can be adapted through a schedule, where $\bar{\beta}_t \to 1$ for $t \to T$ should be guaranteed. As suggested in Ref. [39] we choose a linear increase with $\beta_1 = 10^{-7} T$ and $\beta_T = 2 \cdot 10^{-5} T$.

As a first step towards reversing the forward diffusion, we apply Bayes' theorem on each slice defined in Eq. (3.22),

$$p(x_{t-1}|x_t) = \frac{p(x_t|x_{t-1}) p(x_{t-1})}{p(x_t)} \; . \tag{3.24}$$

However, a closed-form expression for $p(x_t)$ only exists if conditioned on x_0, as given in Eq. (3.23). Using $p(x_t|x_{t-1}, x_0) = p(x_t|x_{t-1})$ we can instead compute the conditioned forward posterior as a Gaussian

$$\begin{aligned} p(x_{t-1}|x_t, x_0) &= \frac{p(x_t|x_{t-1}) p(x_{t-1}|x_0)}{p(x_t|x_0)} = \mathcal{N}(x_{t-1}; \hat{\mu}_t(x_t, x_0), \hat{\beta}_t) \\ \text{with} \quad \hat{\mu}(x_t, x_0) &= \frac{\sqrt{1-\bar{\beta}_{t-1}} \beta_t}{\bar{\beta}_t} x_0 + \frac{\sqrt{1-\beta_t} \bar{\beta}_{t-1}}{\bar{\beta}_t} x_t \quad \text{and} \quad \hat{\beta}_t = \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t} \beta_t \; . \end{aligned} \tag{3.25}$$

The actual reverse process starts with Gaussian noise and gradually transforms it into the phase-space distribution through the same discrete steps as Eq. (3.22), without knowing x_0 a priori. The corresponding generative network needs to approximate Eq. (3.24) for each step. We start by defining our modeled phase-space distribution

$$p_\theta(x_0|\theta) = \int dx_1 ... dx_T \; p(x_0, ..., x_T|\theta) \; , \tag{3.26}$$

and assume that the joint probability is again given by a chain of independent Gaussians,

$$\begin{aligned} p(x_0, ..., x_T|\theta) &= p_{\text{latent}}(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \\ \text{with} \quad p_\theta(x_{t-1}|x_t) &= \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2(x_t, t)) \; . \end{aligned} \tag{3.27}$$

Here, μ_θ and σ_θ are learnable parameters describing the individual conditional probability slices $x_t \to x_{t-1}$. It turns out that in practice we can fix $\sigma_\theta^2(x_t, t) \to \sigma_t^2$ [39]. We will see that the advantage of the discrete diffusion model is that we can compare a Gaussian posterior, Eq. (3.25), with a reverse, learned Gaussian in Eq. (3.27) for each step.

Loss function

Ideally, we want to train our model by maximizing the posterior $p_\theta(\theta|x_0)$, however, this is not tractable. Using Bayes' theorem and dropping regularization and normalization terms this is equivalent to minimizing the corresponding negative log likelihood in Eqs.(3.26) and (3.27),

$$\begin{aligned}
&\Big\langle - \log p_\theta(x_0|\theta) \Big\rangle_{p_\text{data}} \\
&= - \int dx_0 \; p_\text{data}(x_0) \; \log \left(\int dx_1 ... dx_T \; p_\text{latent}(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t) \right) \\
&= - \int dx_0 \; p_\text{data}(x_0) \; \log \left(\int dx_1 ... dx_T \; p_\text{latent}(x_T) p(x_1, ..., x_T|x_0) \prod_{t=1}^T \frac{p_\theta(x_{t-1}|x_t)}{p(x_t|x_{t-1})} \right) \\
&= - \int dx_0 \; p_\text{data}(x_0) \; \log \Big\langle p_\text{latent}(x_T) \prod_{t=1}^T \frac{p_\theta(x_{t-1}|x_t)}{p(x_t|x_{t-1})} \Big\rangle_{p(x_1,...,x_T|x_0)} \qquad (3.28)
\end{aligned}$$

In the first step, we insert a one into our loss function by dividing Eq. (3.22) with itself. Using Jensen's inequality $f(\langle x \rangle) \leq \langle f(x) \rangle$ for convex functions we find

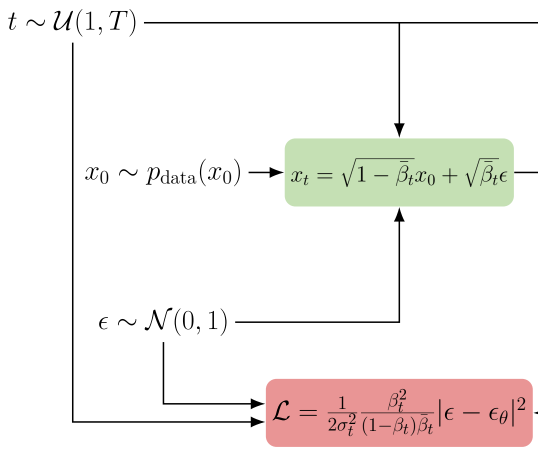
$$\begin{aligned}
\Big\langle - \log p_\theta(x_0|\theta) \Big\rangle_{p_\text{data}} &\leq - \int dx_0 \; p_\text{data}(x_0) \; \left\langle \log \left(p_\text{latent}(x_T) \prod_{t=1}^T \frac{p_\theta(x_{t-1}|x_t)}{p(x_t|x_{t-1})} \right) \right\rangle_{p(x_1,...,x_T|x_0)} \\
&= - \left\langle \log \left(p_\text{latent}(x_T) \prod_{t=1}^T \frac{p_\theta(x_{t-1}|x_t)}{p(x_t|x_{t-1})} \right) \right\rangle_{p(x_0,...,x_T)} \\
&= \left\langle - \log p_\text{latent}(x_T) - \sum_{t=2}^T \log \frac{p_\theta(x_{t-1}|x_t)}{p(x_t|x_{t-1})} - \log \frac{p_\theta(x_0|x_1)}{p(x_1|x_0)} \right\rangle_{p(x_0,...,x_T)} \\
&\equiv \mathcal{L}_\text{DDPM} \; . \qquad (3.29)
\end{aligned}$$

As suggested above, we would like to compare each intermediate learned latent distribution $p_\theta(x_{t-1}|x_t)$ to the real posterior distribution $p(x_{t-1}|x_t, x_0)$ of the forward process. To reverse the ordering of the forward slice we use Bayes' theorem,

$$\begin{aligned}
\mathcal{L}_\text{DDPM} &= \left\langle - \log p_\text{latent}(x_T) - \sum_{t=2}^T \log \frac{p_\theta(x_{t-1}|x_t) p(x_{t-1}|x_0)}{p(x_{t-1}|x_t, x_0) p(x_t|x_0)} - \log \frac{p_\theta(x_0|x_1)}{p(x_1|x_0)} \right\rangle_{p(x_0,...,x_T)} \\
&= \left\langle - \log p_\text{latent}(x_T) - \sum_{t=2}^T \log \frac{p_\theta(x_{t-1}|x_t)}{p(x_{t-1}|x_t, x_0)} - \log \frac{p(x_1|x_0)}{p(x_T|x_0)} - \log \frac{p_\theta(x_0|x_1)}{p(x_1|x_0)} \right\rangle_{p(x_0,...,x_T)} \\
&= \left\langle - \log \frac{p_\text{latent}(x_T)}{p(x_T|x_0)} - \sum_{t=2}^T \log \frac{p_\theta(x_{t-1}|x_t)}{p(x_{t-1}|x_t, x_0)} - \log p_\theta(x_0|x_1) \right\rangle_{p(x_0,...,x_T)} \\
&= \sum_{t=2}^T \Big\langle \text{KL}[p(x_{t-1}|x_t, x_0), p_\theta(x_{t-1}|x_t)] \Big\rangle_{p(x_0,x_t)} + \Big\langle - \log p_\theta(x_0|x_1) \Big\rangle_{p(x_0,...,x_T)} + \text{const} \\
&\approx \sum_{t=2}^T \Big\langle \text{KL}[p(x_{t-1}|x_t, x_0), p_\theta(x_{t-1}|x_t)] \Big\rangle_{p(x_0,x_t)} \qquad (3.30)
\end{aligned}$$

Now, the KL-divergence compares the forward Gaussian step of Eq. (3.25) with the reverse, learned Gaussian in Eq. (3.27). The second sampled term will always be negligible compared to the first $T-1$ terms. The KL-divergence between two Gaussian, with means $\mu_\theta(x_t, t)$ and $\hat{\mu}(x_t, x_0)$ and standard deviations σ_t^2 and $\hat{\beta}_t$, has the compact form

$$\mathcal{L}_\text{DDPM} = \sum_{t=2}^T \Big\langle \frac{1}{2\sigma_t^2} \left| \hat{\mu} - \mu_\theta \right|^2 \Big\rangle_{p(x_0,x_t)} + \text{const}. \qquad (3.31)$$



$t \sim \mathcal{U}(1,T)$
$x_0 \sim p_{\text{data}}(x_0)$
$x_t = \sqrt{1-\bar\beta_t}x_0 + \sqrt{\bar\beta_t}\epsilon$
DDPM
ϵ_θ
$\epsilon \sim \mathcal{N}(0,1)$
$\mathcal{L} = \frac{1}{2\sigma_t^2}\frac{\beta_t^2}{(1-\beta_t)\bar\beta_t}|\epsilon-\epsilon_\theta|^2$

Figure 3.1: DDPM training algorithm, following Ref. [39], with the loss derived in Eq. (3.35).

This implies that μ_θ approximates $\hat\mu$. The sampling follows $p(x_0, x_t) = p(x_t|x_0)\, p_{\text{data}}(x_0)$. We numerically evaluate this loss using the reparametrization trick on Eq. (3.23)

$$\begin{aligned} & x_t(x_0,\epsilon) = \sqrt{1-\bar\beta_t}x_0 + \sqrt{\bar\beta_t}\epsilon \qquad \text{with} \qquad \epsilon \sim \mathcal{N}(0,1) \\ \Leftrightarrow \qquad & x_0(x_t,\epsilon) = \frac{1}{\sqrt{1-\bar\beta_t}}\left(x_t - \sqrt{\bar\beta_t}\epsilon\right) . \end{aligned} \tag{3.32}$$

These expressions provide a closed form for $\hat\mu(x_t, x_0)$, but in terms of x_t and ϵ,

$$\hat\mu(x_t,\epsilon) = \frac{1}{\sqrt{1-\beta_t}}\left(x_t(x_0,\epsilon) - \frac{\beta_t}{\sqrt{\bar\beta_t}}\epsilon\right) . \tag{3.33}$$

For the reverse process we choose the same parametrization, but with a learned $\epsilon_\theta(x_t,t)$,

$$\mu_\theta(x_t,t) \equiv \hat\mu(x_t,\epsilon_\theta) = \frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{\bar\beta_t}}\epsilon_\theta(x_t,t)\right) . \tag{3.34}$$

Inserting both expressions into Eq. (3.31) gives us

$$\mathcal{L}_{\text{DDPM}} = \sum_{t=2}^{T}\left\langle \frac{1}{2\sigma_t^2}\frac{\beta_t^2}{(1-\beta_t)\bar\beta_t}\left|\epsilon - \epsilon_\theta\left(\sqrt{1-\bar\beta_t}x_0 + \sqrt{\bar\beta_t}\epsilon, t\right)\right|^2\right\rangle_{x_0\sim p_{\text{data}},\epsilon\sim\mathcal{N}(0,1)} . \tag{3.35}$$

The sum over t can be evaluated numerically as a sampling. We chose our model variance $\sigma_t^2 \equiv \hat\beta_t$ to follow our true variance. Often, the prefactor in this form for the loss is neglected in the training, but as we need a likelihood loss for the Bayesian setup and no drop in performance was observed, we keep it.

The DDPM model belongs to the broad class of score-based models, and Eq. (3.31) can also be reformulated for the model to predict the score $s(x_t,t) = \nabla_{x_t}\log p(x_t)$ of our latent space at time t. It can be shown that $s_\theta(x_t,t) = -\epsilon_\theta(x_t,t)/\sigma_t$ [73].

Training and sampling

The training algorithm for the DDPM is illustrated in Fig. 3.1. For a given phase-space point $x_0 \sim p_{\text{data}}(x_0)$ drawn from our true phase space distribution we draw a time step $t \sim \mathcal{U}(1, T)$ from a uniform distribution as well as Gaussian noise $\epsilon \sim \mathcal{N}(0, 1)$ at each iteration. Given Eq. (3.33) we can then calculate our diffused data after t time steps x_t, which is fed to the DDPM network together with our condition t. The network encodes ϵ_θ and we compare this network prediction with the true Gaussian noise ϵ multiplied by a t-dependent constant as given in the likelihood loss of Eq. (3.35). Note that we want to ensure that our network sees as many different time steps t for many different phase-space points x_0 as necessary to learn the step-wise reversed diffusion process.

The sampling algorithm for the DDPM is shown in Fig. 3.2. We start by feeding our network our final timestep T and $x_T \sim p_{\text{latent}}(x_T)$ drawn from our Gaussian latent space distribution. With the predicted ϵ_θ and drawn Gaussian noise $z_{T-1} \sim \mathcal{N}(0, 1)$ we can then calculate x_{T-1}, which is a slightly less diffused version of x_T. This procedure is repeated until reaching our phase-space and computing x_0, where no additional Gaussian noise is added. Note that during sampling the model needs to predict ϵ_θ T times, making the sampling process slower than for other generative networks.

Likelihood extraction

To calculate the model likelihood we can use Eq. (3.26) or its sampled estimate,

$$p_\theta(x_0|\theta) = \Big\langle p_\theta(x_0|x_1) \Big\rangle_{p(x_1,\dots,x_T|\theta)} \, , \tag{3.36}$$

but this is very inefficient. The problem is that $p_\theta(x_0|x_1)$ is a narrow distribution, essentially zero for almost all sampled x_1. We can improve the efficiency by importance sampling and use instead

$$\begin{aligned} p_\theta(x_0|\theta) &= \left\langle \frac{p(x_1,\dots,x_T|\theta)}{p(x_1,\dots,x_T|x_0)} p_\theta(x_0|x_1) \right\rangle_{p(x_1,\dots,x_T|x_0)} \\ &= \left\langle \frac{p(x_0,\dots,x_T|\theta)}{p(x_1,\dots,x_T|x_0)} \right\rangle_{p(x_1,\dots,x_T|x_0)} \, . \end{aligned} \tag{3.37}$$

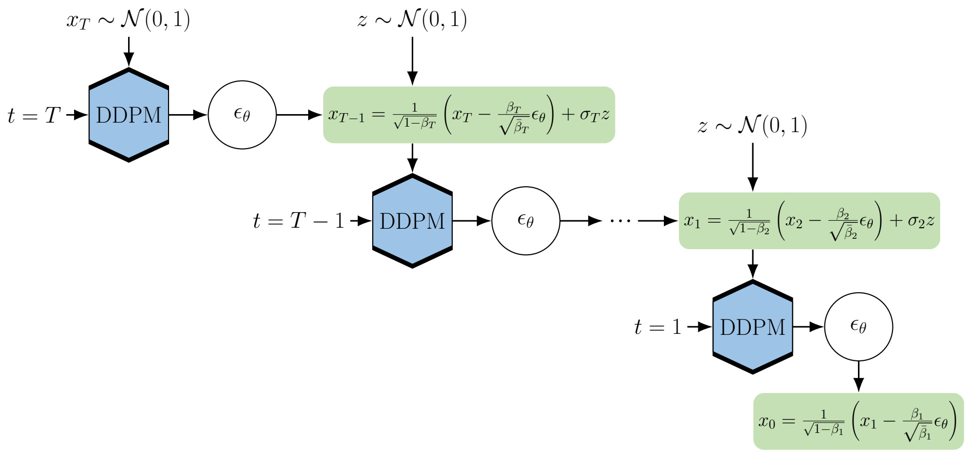

$x_T \sim \mathcal{N}(0,1)$
$z \sim \mathcal{N}(0,1)$
$t = T$
DDPM
ϵ_θ
$x_{T-1} = \frac{1}{\sqrt{1-\beta_T}}\left(x_T - \frac{\beta_T}{\sqrt{\bar\beta_T}}\epsilon_\theta\right) + \sigma_T z$
$z \sim \mathcal{N}(0,1)$
$t = T-1$
DDPM
ϵ_θ
$\cdots$
$x_1 = \frac{1}{\sqrt{1-\beta_2}}\left(x_2 - \frac{\beta_2}{\sqrt{\bar\beta_2}}\epsilon_\theta\right) + \sigma_2 z$
$t = 1$
DDPM
ϵ_θ
$x_0 = \frac{1}{\sqrt{1-\beta_1}}\left(x_1 - \frac{\beta_1}{\sqrt{\bar\beta_1}}\epsilon_\theta\right)$

Figure 3.2: DDPM sampling algorithm, following Ref. [39].

This samples a diffusion process starting from x_0 and into the latent space, meaning that it represents a likely forward and backward path. This means the integrand should not just be zero most of the time.

Bayesian DDPM

The derivation of Eq. (3.18) can be easily extended to the density estimation step of a generative networks, in the same way as for the Bayesian Invertible Neural Networks (INNs) [66]. The Bayesian DDPM loss follows from the deterministic likelihood loss in Eqs.(3.29) and (3.35) by adding a sampling over $\theta \sim q(\theta)$ and the regularization term,

$$\mathcal{L}_{\text{B-DDPM}} = \Big\langle \mathcal{L}_{\text{DDPM}} \Big\rangle_{\theta \sim q} + \text{KL}[q(\theta), p(\theta)] \; . \tag{3.38}$$

3.4.2 Conditional Flow Matching

Architecture

Next, we study Conditional Flow Matching (CFM) [40, 74, 75]. Like the DDPM, it uses a time evolution to transform phase space samples into noise, so the reverse direction can generate samples as outlined in Eq. (3.21). Instead of a discrete chain of conditional probabilities, the time evolution of samples in the CFM framework follows a continuous ordinary differential equation (ODE)

$$\frac{dx(t)}{dt} = v(x(t), t) \qquad \text{with} \qquad x(t=0) = x_0 \; , \tag{3.39}$$

where $v(x(t), t)$ is called the velocity field of the process. This velocity field can be linked to a probability density $p(x, t)$ normalized over x with the continuity equation

$$\frac{\partial p(x,t)}{\partial t} + \nabla_x \left[p(x,t) v(x,t) \right] = 0 \; . \tag{3.40}$$

These two equations are equivalent in the sense that for a given probability density path $p(x, t)$ any velocity field $v(x, t)$ describing the sample-wise evolution Eq. (3.39) will be a solution of Eq. (3.40), and vice versa. Our generative model employs $p(x, t)$ to transforms a phase space distribution into a Gaussian latent distribution

$$p(x,t) \to \begin{cases} p_{\text{data}}(x) & t \to 0 \\ p_{\text{latent}}(x) = \mathcal{N}(x; 0, 1) & t \to 1 \; . \end{cases} \tag{3.41}$$

The associated velocity field will allow us to generate samples by integrating the ODE of Eq. (3.39) from $t = 1$ to $t = 0$.

As for the DDPM, we start with a diffusion direction. We define the time evolution from a phase space point x_0 to the standard Gaussian as

$$x(t|x_0) = (1-t)x_0 + t\epsilon \to \begin{cases} x_0 & t \to 0 \\ \epsilon \sim \mathcal{N}(0,1) & t \to 1 \; , \end{cases} \tag{3.42}$$

following a simple linear trajectory [74]. For given x_0 we can generate $x(t|x_0)$ by sampling

$$p(x,t|x_0) = \mathcal{N}(x;(1-t)x_0,t) \; . \tag{3.43}$$

This conditional time evolution is similar to the DDPM case in Eq. (3.23), and it gives us the complete probability path

$$p(x,t) = \int dx_0 \; p(x,t|x_0) \; p_\text{data}(x_0) \; . \tag{3.44}$$

It fulfills the boundary conditions in Eq. (3.41),

$$\begin{aligned} p(x,0) &= \int dx_0 \; p(x,0|x_0) \; p_\text{data}(x_0) = \int dx_0 \; \delta(x-x_0) \; p_\text{data}(x_0) = p_\text{data}(x) \\ p(x,1) &= \int dx_0 \; p(x,1|x_0) \; p_\text{data}(x_0) = \mathcal{N}(x;0,1) \int dx_0 \; p_\text{data}(x_0) = \mathcal{N}(x;0,1) \; . \end{aligned} \tag{3.45}$$

From this probability density path we need to extract the velocity field. We start with the conditional velocity, associated with $p(x,t|x_0)$, and combine Eq. (3.39) and (3.42) to

$$v(x(t|x_0),t|x_0) = \frac{d}{dt}\left[(1-t)x_0 + t\epsilon\right] = -x_0 + \epsilon \; . \tag{3.46}$$

The linear trajectory leads to a time-constant velocity, which solves the continuity equation for $p(x,t|x_0)$ by construction. We exploit this fact to find the unconditional $v(x,t)$

$$\begin{aligned} \frac{\partial p(x,t)}{\partial t} &= \int dx_0 \; \frac{\partial p(x,t|x_0)}{\partial t} \; p_\text{data}(x_0) \\ &= -\int dx_0 \; \nabla_x \left[v(x,t|x_0)p(x,t|x_0)\right] \; p_\text{data}(x_0) \\ &= -\nabla_x \left[p(x,t) \int dx_0 \; \frac{v(x,t|x_0)p(x,t|x_0)p_\text{data}(x_0)}{p(x,t)}\right] \\ &= -\nabla_x \left[p(x,t)v(x,t)\right] \; , \end{aligned} \tag{3.47}$$

by defining

$$v(x,t) = \int dx_0 \; \frac{v(x,t|x_0)p(x,t|x_0)p_\text{data}(x_0)}{p(x,t)} \; . \tag{3.48}$$

Whereas the conditional velocity in Eq. (3.46) describes a trajectory between a normal distributed and a phase space sample x_0 that is specified in advance, the aggregated velocity in Eq. (3.48) can evolve samples from p_data to p_latent and vice versa.
Like the DDPM model, the CFM model can be linked to score-based diffusion models, [75] derive a general relation between the velocity field and the score of a diffusion process that for our linear trajectory reduces to $s(x,t) = -\frac{1}{t}(x + (1-t)v(x,t))$.

Loss function

Encoding the velocity field in Eq. (3.48) is a simple regression task, $v(x,t) \approx v_\theta(x,t)$. The straightforward choice for the loss is the mean squared error,

$$\Big\langle [v_\theta(x,t) - v(x,t)]^2 \Big\rangle_{t,x\sim p(x,t)} = = \Big\langle v_\theta(x,t)^2 \Big\rangle_{t,x\sim p(x,t)} - \Big\langle 2v_\theta(x,t)v(x,t) \Big\rangle_{t,x\sim p(x,t)} + \text{const} \,, \tag{3.49}$$

where the time is sampled uniformly over $t \in [0,1]$. While we would like to sample x from the probability path given in Eq. (3.44) and learn the velocity field given in Eq. (3.48), neither of these is tractable. However, it is easy to sample from the conditional path in Eq. (3.43) and calculate the conditional velocity in Eq. (3.46). We rewrite the above loss in terms of the conditional quantities, so the first term becomes

$$\begin{aligned}
\Big\langle v_\theta(x,t)^2 \Big\rangle_{t,x\sim p(x,t)} &= \Big\langle \int dx v_\theta(x,t)^2 \int dx_0 \; p(x,t|x_0) p_\text{data}(x_0) \Big\rangle_t \\
&= \Big\langle v_\theta(x,t)^2 \Big\rangle_{t,x_0\sim p_\text{data},x\sim p(x,t|x_0)} \\
&= \Big\langle v_\theta(x(t|x_0),t)^2 \Big\rangle_{t,x_0\sim p_\text{data},\epsilon}
\end{aligned} \tag{3.50}$$

Using Eq. (3.48) we can rewrite the second loss term as

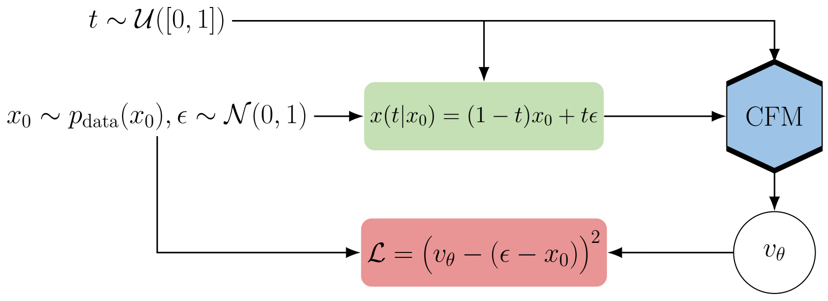
$$\begin{aligned}
-2\Big\langle v_\theta(x,t)v(x,t) \Big\rangle_{t,x\sim p(x,t)} &= -2\Big\langle \int dx \; p(x,t) v_\theta(x,t) \; \frac{\int dx_0 v(x,t|x_0)p(x,t|x_0)p_\text{data}(x_0)}{p(x,t)} \Big\rangle_t \\
&= -2\Big\langle \int dx dx_0 \; v_\theta(x,t) \; v(x,t|x_0) \; p(x,t|x_0) \; p_\text{data}(x_0) \Big\rangle_t \\
&= -2\Big\langle v_\theta(x,t) \; v(x,t|x_0) \Big\rangle_{t,x_0\sim p_\text{data},x\sim p(x,t|x_0)} \\
&= -2\Big\langle v_\theta(x(t|x_0),t) \; v(x(t|x_0),t|x_0) \Big\rangle_{t,x_0\sim p_\text{data},\epsilon} \,.
\end{aligned} \tag{3.51}$$

The (conditional) Flow Matching loss of Eq. (3.49) then becomes

$$\begin{aligned}
\mathcal{L}_\text{CFM} &= \Big\langle [v_\theta(x(t|x_0),t) - v(x(t|x_0),t|x_0)]^2 \Big\rangle_{t,x_0\sim p_\text{data},\epsilon} \\
&= \Big\langle \Big[v_\theta(x(t|x_0),t) - \frac{dx(t|x_0))}{dt} \Big]^2 \Big\rangle_{t,x_0\sim p_\text{data},\epsilon} \\
&= \Big\langle [v_\theta((1-t)x_0 + t\epsilon, t) - (\epsilon - x_0)]^2 \Big\rangle_{t,x_0\sim p_\text{data},\epsilon} \,.
\end{aligned} \tag{3.52}$$

Training and Sampling

The CFM training is illustrated in Fig. 3.3. At each iteration we sample a data point $x_0 \sim p_\text{data}(x_0)$ and a normally-distributed $\epsilon \sim \mathcal{N}(0,1)$ as starting and end points of a trajectory, as well as a time $t \sim \mathcal{U}([0,1])$. We then compute $x(t|x_0)$ following Eq. (3.42) and the associated conditional velocity $v(x(t|x_0),t|x_0)$ following Eq. (3.46). The point $x(t|x_0)$ and the time t are passed to a neural network which encodes the conditional velocity field $v_\theta(x,t) \approx v(x,t|x_0)$. One property of the training algorithm is that the same network input, a time t and a position $x(t|x_0)$, can be produced by many different trajectories, each with a different conditional velocity. While the network training is



$t \sim \mathcal{U}([0,1])$
$x_0 \sim p_{\text{data}}(x_0), \epsilon \sim \mathcal{N}(0,1)$
$x(t|x_0) = (1-t)x_0 + t\epsilon$
CFM
v_θ
$\mathcal{L} = \left(v_\theta - (\epsilon - x_0)\right)^2$

Figure 3.3: CFM training algorithm, with the loss derived in Eq. (3.52).

based on a wide range of possible trajectories, the CFM loss in Eq. (3.52) ensures that sampling over many trajectories returns a well-defined velocity field.

Once the CFM model is trained, the generation of new samples is straightforward. We start by drawing a sample from the latent distribution $x_1 \sim p_{\text{latent}} = \mathcal{N}(0,1)$ and calculate its time evolution by numerically solving the ODE backwards in time from $t = 1$ to $t = 0$

$$\begin{aligned} & \frac{d}{dt} x(t) = v_\theta(x(t), t) \qquad \text{with} \quad x_1 = x(t=1) \\ \Rightarrow \qquad & x_0 = x_1 - \int_0^1 v_\theta(x,t) dt \equiv G_\theta(x_1) \ , \end{aligned} \tag{3.53}$$

Under mild regularity assumptions this solution defines a bijective transformation between the latent space sample and the phase space sample $G_\theta(x_1)$.

Likelihood extraction

The CFM model also allows to calculate phase space likelihoods. Making use of the continuity equation we can write

$$\begin{aligned} \frac{dp(x,t)}{dt} &= \frac{\partial p(x,t)}{\partial t} + \nabla_x p(x,t)\ v(x,t) \\ &= \frac{\partial p(x,t)}{\partial t} + \nabla_x \left[p(x,t) v(x,t)\right] - p(x,t) \nabla_x v(x,t) \\ &= -p(x,t) \nabla_x v(x,t) \ . \end{aligned} \tag{3.54}$$

Its solution is

$$\frac{p(x_1,1)}{p(x_0,0)} \equiv \frac{p_{\text{latent}}(G_\theta^{-1}(x_0))}{p_\theta(x_0|\theta)} = \exp\left(-\int_0^1 dt \nabla_x v_\theta(x(t),t)\right) \ , \tag{3.55}$$

and we can write in the usual notation [72]

$$\begin{aligned} & p_\theta(x_0|\theta) = p_{\text{latent}}(G_\theta^{-1}(x_0)) \left|\det \frac{\partial G_\theta^{-1}(x_0)}{\partial x_0}\right| \\ \Rightarrow \qquad & \left|\det \frac{\partial G_\theta^{-1}(x_0)}{\partial x_0}\right| = \exp\left(\int_0^1 dt \nabla_x v_\theta(x(t),t)\right) \ . \end{aligned} \tag{3.56}$$

Calculating the Jacobian requires integrating over the divergence of the learned velocity field. This divergence can be calculated using automatic differentiation approximately as fast as n network calls, where n is the data dimensionality.

Bayesian CFM

Finally, we also turn the CFM into a Bayesian generative model, to account for the uncertainties in the underlying density estimation [66]. From the Bayesian DDPM we know that this can be achieved by promoting the network weights from deterministic values to, for instance, Gaussian distributions and using variational approximation for the training [67–70]. For the Bayesian DDPM the loss is a sum of the likelihood loss and a KL-divergence regularization, Eq. (3.38). Unfortunately, the CFM loss in Eq. (3.52) is not a likelihood loss. To construct a Bayesian CFM loss we therefore combine it with Bayesian network layers and a free KL-regularization,

$$\mathcal{L}_{\text{B-CFM}} = \Big\langle \mathcal{L}_{\text{CFM}} \Big\rangle_{\theta \sim q(\theta)} + c \, \text{KL}[q(\theta), p(\theta)]. \tag{3.57}$$

While for a likelihood loss the factor c is fixed by Bayes' theorem, in the CFM case it is a free hyperparameter. We find that the network predictions and their associated uncertainties are very stable when varying it over several orders of magnitude.

3.5 Distribution Mapping

The content of this section was finalized in collaboration with Anja Butter, Sascha Diefenbacher, Nathan Huetsch, Vinicius Mikuni, Benjamin Nachman and Tilman Plehn. It is adapted from Ref. [4].

So far we have only looked into the scenario of Eq. (3.20), however diffusion networks offer the possibility of sampling a phase space distribution starting from any given distribution, not just from Gaussians or other standard distributions [2,76–78]. To derive this property, we review in some detail how this feature can be realized for stochastic conditional sampling. We start by repeating some aspects from the previous discussion to introduce the formalism.

3.5.1 Distribution to noise

We start by designing a process $p(x,t)$ that transforms a general data distribution into a Gaussian latent distribution,

$$p(x,t) = \begin{cases} p_{\text{data}}(x) & t = 0 \\ p_{\text{latent}}(x) = \mathcal{N}(0,1) & t = 1 \;. \end{cases} \tag{3.58}$$

This can be done using the stochastic differential equation (SDE)

$$dx = f(x,t)\; dt + g(t)\; dW \;. \tag{3.59}$$

Here $f(x,t)$ is the so-called drift coefficient, describing the deterministic part of the time evolution. The diffusion coefficient $g(t)$ describes the strength of the noising process, and W is a standard Wiener process, dW a noise infinitesimal. The connection between the evolving density in Eq. (3.58) and the trajectories in Eq. (3.59) is given by the Fokker-Planck equation (FPE)

$$\frac{\partial p(x,t)}{\partial t} = -\nabla_x[f(x,t)p(x,t)] + \frac{g(t)^2}{2}\nabla_x^2 p(x,t) \; . \tag{3.60}$$

For $g(t) = 0$ the SDE reduces to an ordinary differential equation (ODE), and the FPE to the usual continuity equation. Unlike ODEs, SDEs are not time-symmetric. This is because adding noise to a system is fundamentally different from removing noise. It can be shown that the time-reversal of Eq. (3.59) is given by another diffusion SDE [79],

$$dx = \left[f(x,t) - g(t)^2 \, \nabla_x \log p(x,t)\right] \, dt + g(t) \, d\bar{W} \; . \tag{3.61}$$

where $d\bar{W}$ is the noise infinitesimal corresponding to the reverse Wiener process. The new and unknown element is the score function

$$s(x,t) = \nabla_x \log p(x,t) \; , \tag{3.62}$$

where $p(x,t)$ is the solution to the forward and reverse SDEs. If we know the score function, we can use numerical SDE solvers to propagate samples backward in time.

Forward process

Diffusion generative networks usually define the latent space to be a standard Gaussian $\mathcal{N}(0,1)$. We can construct the forward process Eq. (3.59) by simplifying the drift to be linear in x, i.e. $f(x,t) = xf(t)$. Now the SDE and the FPE read

$$\begin{aligned} dx &= x \, f(t) \, dt + g(t) \, dW \; . \\ \frac{\partial p(x,t)}{\partial t} &= -f(t)p(x,t) - xf(t)\nabla_x p(x,t) + \frac{g(t)^2}{2}\nabla_x^2 p(x,t) \; . \end{aligned} \tag{3.63}$$

In this case, we can analytically derive the solution of the FPE for given $f(t)$ and $g(t)$. We make the ansatz that the time evolution starting from an event $x_0 \sim p_{\text{data}}$ is a Gaussian

$$\begin{aligned} & p(x,t|x_0) = \mathcal{N}(x|\mu(t),\sigma(t)) = \frac{1}{\sqrt{2\pi}\sigma(t)} \exp\left(-\frac{(x-\mu(t))^2}{2\sigma(t)^2}\right) \\ \Leftrightarrow \qquad & x(t|x_0) = \mu(t) + \sigma(t)\epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0,1) \; , \end{aligned} \tag{3.64}$$

with time dependent mean $\mu(t)$ and standard deviation $\sigma(t)$. Using this ansatz in the FPE Eq. (3.63), we obtain

$$\begin{aligned} \frac{x-\mu}{\sigma^2}\frac{\partial\mu}{\partial t} + \frac{(x-\mu)^2}{\sigma^3}\frac{\partial\sigma}{\partial t} - \frac{1}{\sigma}\frac{\partial\sigma}{\partial t} &= f(t)\left(x\frac{x-\mu}{\sigma^2} - 1\right) + \frac{g(t)^2}{2}\left(\frac{(x-\mu)^2}{\sigma^4} - \frac{1}{\sigma^2}\right) \\ &= f(t)\frac{(x-\mu)^2 + \mu(x-\mu) - \sigma^2}{\sigma^2} + g(t)^2\frac{(x-\mu)^2 - \sigma^2}{2\sigma^4} \; . \end{aligned} \tag{3.65}$$

Sorting this equation by powers of $(x-\mu)$ and comparing coefficients we find relations between $\mu(t), \sigma(t)$ and $f(t), g(t)$,

$$\frac{\partial\mu(t)}{\partial t} = f(t)\mu(t)$$

$$\frac{\partial \sigma(t)}{\partial t} = f(t)\sigma(t) + \frac{g(t)^2}{2\sigma(t)} \, . \tag{3.66}$$

The solutions of those two differential equations with initial conditions $\sigma(0) = 0$ and $\mu(0) = x_0$ are

$$\begin{aligned} \mu(t) &= x_0 \alpha(t) \\ \sigma(t) &= \alpha(t) \left[\int_0^t dt' \frac{g(t')^2}{\alpha(t')^2} \right]^{1/2} \qquad \text{with} \qquad \alpha(t) = \exp \int_0^t dt' f(t') \, . \end{aligned} \tag{3.67}$$

If these equations are fulfilled, the Gaussian ansatz is the solution to the FPE. This gives us a solution for general $f(t), g(t)$. However, only if the boundary conditions $\alpha(1) = 0$ and $\sigma(1) = 1$ are fulfilled, the full unconditional density follows

$$\begin{aligned} p(x,0) &= \int dx_0 \; p(x,0|x_0) \; p_\text{data}(x_0) = \int dx_0 \; \delta(x - x_0) \; p_\text{data}(x_0) = p_\text{data}(x) \\ p(x,1) &= \int dx_0 \; p(x,1|x_0) \; p_\text{data}(x_0) = \mathcal{N}(x;0,1) \int dx_0 \; p_\text{data}(x_0) = \mathcal{N}(0,1) \, , \end{aligned} \tag{3.68}$$

as specified in Eq. (3.58).

Relation to CFM

Equations (3.64) and (3.67) describe the mathematics behind all generative diffusion networks. Conditional flow matching (CFM) [40], which we introduced in Section 3.4.2, can be derived from this formalism. First, we use the fact that for any diffusion SDE there exists an associated ODE that encodes the same time-dependent density $p(x,t)$. It can be derived by rewriting the FPE (3.60) as

$$\begin{aligned} \frac{\partial p(x,t)}{\partial t} &= -\nabla_x \left[\left(f(x,t) - \frac{1}{2} g(t)^2 \nabla_x \log p(x,t) \right) p(x,t) \right] \\ &= -\nabla_x (v(x,t) p(x,t)) \\ \text{with} \quad v(x,t) &= f(x,t) - \frac{1}{2} g(t)^2 \nabla_x \log p(x,t) \, . \end{aligned} \tag{3.69}$$

This continuity equation corresponds to the ODE

$$dx = v(x,t) \; dt \, . \tag{3.70}$$

The deterministic (ODE) and stochastic (SDE) processes are equivalent in the sense that they have the same density solution $p(x,t)$. The difference between the SDE drift $f(x,t)$ and the ODE velocity $v(x,t)$ is that the former can be hand-crafted such that the forward SDE transports from the data to the latent distribution. To generate samples the score function $s(x,t)$ is also required. The velocity field combines the forward drift and the score function of the underlying process into one time-symmetric description. For known $f(x,t)$ and $g(t)$ the velocity and the score functions can be converted into each other.

CFMs work directly with the velocity field v of the ODE, the underlying SDE is not explicitly constructed. Instead, the trajectories Eq. (3.64) are used to define a forward process from data to noise. We can set $\alpha(t) = (1 - t)$ and $\sigma(t) = t$, defining linear trajectories

$$x(t|x_0) = (1-t)x_0 + t\epsilon \tag{3.71}$$

and recover the same formalism as in Eq. (3.42).

3.5.2 Distribution to distribution

We now extend the SDE formalism to two arbitrary distributions. The generative direction starts from the initial $p(x_0)$ and samples the $p(x_1)$. The goal is to find a drift function $f(x,t)$ such that the SDE moves from $x_0 \sim p(x_0)$ at time $t=0$ to $x_1 \sim p(x_1)$ at time $t=1$.

Doob's h-transform

The key ingredient to this generalization is Doob's h-transform [80]. It conditions a given SDE, called the reference process, on a pre-defined final point. The reference process follows an SDE like Eq. (3.59),

$$dx_{\text{ref}} = f(x,t)\; dt + g(t)\; dW \; . \tag{3.72}$$

From $t=0$ to $t=1$ it encodes a time-evolving density $p_{\text{ref}}(x,t)$ for the entire stochastic process, as well as the conditional $p_{\text{ref}}(x,t|x_0)$ describing the stochastic trajectories starting from a specific x_0.

We modify this SDE to guarantee that the endpoint is a pre-defined x_1, by adding a term to the drift function,

$$\begin{aligned} dx &= \Big[f(x,t) + g(t)^2 h(x,t,x_1) \Big] \; dt + g(t) dW \\ \text{with} \quad h(x,t,x_1) &= \nabla_x \log p_{\text{ref}}(x_1, t=1|x) \; . \end{aligned} \tag{3.73}$$

The Doob's h-transform function depends on the current state of the SDE $x(t)$ at time t and on the specified final point x_1. The density $p_{\text{ref}}(x_1, t=1|x)$ is the transition probability that the reference process reaches x_1 at $t=1$ conditioned on the state $x(t)$ at time t. Including this term in the drift forces the trajectories to walk up the gradient of this density and pushes them towards intermediate states that are more compatible with the desired final state. This way, it corrects the initial SDE by forcing it towards the specified x_1.

We note that h depends on the reference process through $p_{\text{ref}}(x_1, t=1|x)$. This correction adapts to the chosen initial SDE. Different initial values $f(x,t)$ and $g(t)$ lead to a different correction from the Doob's h-transform, but eventually arrive in the specified x_1. This means we can simplify the reference process into a pure noising,

$$f(x,t) = 0 \qquad\qquad \Rightarrow \qquad dx^{\text{ref}} = g(t)\; dW \; . \tag{3.74}$$

For this choice we can use Eqs.(3.67) and (3.73) to calculate the h-transform

$$\begin{aligned} \alpha_{\text{ref}}(t) &= 1 \\ \sigma_{\text{ref}}(t)^2 &= \int_t^1 dt' g(t')^2 \end{aligned}$$

$$\begin{aligned}
p_{\text{ref}}(x_1, t=1|x) &= \mathcal{N}\Big(x_1; x, \sigma_{\text{ref}}(t)\Big) \propto \exp\left[-\frac{1}{2}\frac{(x_1 - x(t))^2}{\sigma_{\text{ref}}(t)^2}\right] \\
h(x, t, x_1) &= \frac{x_1 - x(t)}{\sigma_{\text{ref}}(t)^2} \ .
\end{aligned} \tag{3.75}$$

We have explicitly constructed a forward SDE

$$dx = \frac{g(t)^2}{\sigma_{\text{ref}}(t)^2}(x(t) - x_1)\ dt + g(t)\ dW \ , \tag{3.76}$$

for which the solution trajectories are guaranteed to end in $x_1 \sim p_1$. According to Eq. (3.60) the underlying probability distribution fulfills

$$\frac{\partial p(x, t|x_1)}{\partial t} = -\nabla_x \frac{g(t)^2 (x(t) - x_1) p(x, t|x_1)}{\sigma_{\text{ref}}(t)^2} + \frac{g(t)^2}{2}\nabla_x^2 p(x, t|x_1) \ . \tag{3.77}$$

Using the same method, we can also describe a reverse process, for which the solution trajectories end in $x_0 \sim p_0$. The reference process

$$d\bar{x}_{\text{ref}} = g(t)\ d\bar{W} \ , \tag{3.78}$$

moves from $t = 1$ to $t = 0$. In complete analogy, it encodes the conditional probability $\bar{p}_{\text{ref}}(\bar{x}(t)|x_1)$ starting from a specific point x_1. Applying the h-transform leads to the SDE

$$\begin{aligned}
d\bar{x} &= \Big[-g(t)^2 \bar{h}(\bar{x}, t, x_0)\Big]\ dt + g(t) d\bar{W} \\
\text{with} \quad \bar{h}(\bar{x}, t, x_0) &= \nabla_{\bar{x}} \log \bar{p}_{\text{ref}}(x_0, t=0|\bar{x}) \ ,
\end{aligned} \tag{3.79}$$

and modifies Eq. (3.75) to

$$\begin{aligned}
\bar{\sigma}_{\text{ref}}(t)^2 &= \int_0^t dt' g(t')^2 \\
\bar{h}(\bar{x}, t, x_0) &= \frac{x_0 - \bar{x}(t)}{\bar{\sigma}_{\text{ref}}(t)^2} \ .
\end{aligned} \tag{3.80}$$

Again we constructed a generating SDE

$$d\bar{x} = \frac{g(t)^2}{\bar{\sigma}_{\text{ref}}(t)^2}(\bar{x}(t) - x_0)\ dt + g(t)\ d\bar{W} \ . \tag{3.81}$$

whose solutions end in x_0 and the underlying probability density follows

$$\frac{\partial \bar{p}(x, t|x_0)}{\partial t} = -\nabla_{\bar{x}} \frac{g(t)^2 (\bar{x}(t) - x_0) \bar{p}(\bar{x}, t|x_0)}{\bar{\sigma}_{\text{ref}}(t)^2} - \frac{g(t)^2}{2}\nabla_{\bar{x}}^2 \bar{p}(\bar{x}, t|x_0) \ . \tag{3.82}$$

So far, we have constructed the forward and the reverse processes independently. We now assume that they are the time-reversal of each other and that the forward and reverse FPEs (3.77) and (3.82) have a common Gaussian solution

$$p(x, t|x_0, x_1) = \bar{p}(\bar{x}, t|x_0, x_1) = \mathcal{N}(x|\mu(t), \sigma(t)) \ . \tag{3.83}$$

Inserting this ansatz into the forward FPE (3.77), we obtain

$$\frac{x-\mu}{\sigma^2}\frac{\partial\mu}{\partial t}+\frac{(x-\mu)^2}{\sigma^3}\frac{\partial\sigma}{\partial t}-\frac{1}{\sigma}\frac{\partial\sigma}{\partial t}=\frac{g^2}{\sigma_\text{ref}^2}\left(\frac{(x_1-x)(x-\mu)}{\sigma^2}+1\right)+\frac{g^2}{2}\left(\frac{(x-\mu)^2}{\sigma^4}-\frac{1}{\sigma^2}\right)\,. \tag{3.84}$$

It is solved by

$$\begin{aligned}\frac{\partial\mu(t)}{\partial t}&=\frac{g(t)^2(x_1-\mu(t))}{\sigma_\text{ref}(t)^2}\\ \frac{\partial\sigma(t)}{\partial t}&=-\frac{g(t)^2\sigma(t)}{\sigma_\text{ref}(t)^2}+\frac{g(t)^2}{2\sigma(t)}\,.\end{aligned} \tag{3.85}$$

From the reverse FPE (3.82) we find the corresponding

$$\begin{aligned}\frac{\partial\mu(t)}{\partial t}&=\frac{g(t)^2(\mu(t)-x_0)}{\bar\sigma_\text{ref}(t)^2}\\ \frac{\partial\sigma(t)}{\partial t}&=\frac{g(t)^2\sigma(t)}{\bar\sigma_\text{ref}(t)^2}-\frac{g(t)^2}{2\sigma(t)}\,.\end{aligned} \tag{3.86}$$

Equating Eq. (3.85) and Eq. (3.86) yields

$$\begin{aligned}\mu(t)&=\frac{\bar\sigma_\text{ref}(t)^2x_1+\sigma_\text{ref}(t)^2x_0}{\bar\sigma_\text{ref}(t)^2+\sigma_\text{ref}(t)^2}\\ \sigma(t)&=\sqrt{\frac{\bar\sigma_\text{ref}(t)^2\sigma_\text{ref}(t)^2}{\bar\sigma_\text{ref}(t)^2+\sigma_\text{ref}(t)^2}}\,.\end{aligned} \tag{3.87}$$

This solves Eq. (3.85) and Eq. (3.86) with the boundary conditions $\sigma(0)=\sigma(1)=0$, $\mu(0)=x_0$ and $\mu(1)=x_1$. Finally, the conditional probability of both processes is given by

$$\begin{aligned}p(x(t),t|x_0,x_1)&=\mathcal{N}\left(x\Big|\frac{\bar\sigma_\text{ref}(t)^2x_1+\sigma_\text{ref}(t)^2x_0}{\bar\sigma_\text{ref}(t)^2+\sigma_\text{ref}(t)^2},\sqrt{\frac{\bar\sigma_\text{ref}(t)^2\sigma_\text{ref}(t)^2}{\bar\sigma_\text{ref}(t)^2+\sigma_\text{ref}(t)^2}}\right)\\ &\propto\mathcal{N}(x|x_0,\bar\sigma_\text{ref})\,\mathcal{N}(x|x_1,\sigma_\text{ref})\,.\end{aligned} \tag{3.88}$$

Loss function

The full unconditional density encoded in the constructed stochastic process is obtained by marginalizing over the conditions.

$$\begin{aligned}p(x,t)&=\int dx_0\,dx_1\,p^\text{train}(x_0,x_1)\,p(x,t|x_0,x_1)\\ &=\int dx_0\,dx_1\,p^\text{train}(x_0,x_1)\,\mathcal{N}(x|\mu(t),\sigma(t))=\begin{cases}p_0(x) & t=0\\ p_1(x) & t=1\,.\end{cases}\end{aligned} \tag{3.89}$$

The joint distribution $p^\text{train}(x_0,x_1)$ is defined by the pairing in the training data, in the case of unpaired data it factorizes to $p^\text{train}(x_0,x_1)=p_0(x_0)p_1(x_1)$. Both limits of the stochastic process are fulfilled independent of the choice of joint distribution. For instance, for unfolding, we can use the pairing between reco-level and gen-level events from the forward simulation.

To construct a generative network, we need to remove the conditions on the two end points. This means we want to find an SDE that encodes the distribution $p(x,t)$, but with a drift function that only depends on the current state of the SDE. We can derive this unconditional drift term similarly to the unconditional CFM-velocity [1,40], starting with the FPE (3.60),

$$\begin{aligned}
\frac{\partial p(x,t)}{\partial t} &= \int dx_0 dx_1 p^{\text{train}}(x_0,x_1) \frac{\partial p(x,t|x_0,x_1)}{\partial t} \\
&= \int dx_0 dx_1 p^{\text{train}}(x_0,x_1) \left[-\nabla_x [f(x,t|x_0,x_1) p(x,t|x_0,x_1)] + \frac{g^2}{2} \nabla_x^2 p(x,t|x_0,x_1) \right] \\
&= -\nabla_x \left[p(x,t) \int dx_0 dx_1 p^{\text{train}}(x_0,x_1) \frac{f(x,t|x_0,x_1) p(x,t|x_0,x_1)}{p(x,t)} \right] \\
&\quad + \frac{g^2}{2} \nabla_x^2 \int dx_0 dx_1 p^{\text{train}}(x_0,x_1) p(x,t|x_0,x_1) \\
&= -\nabla_x [p(x,t) f(x,t)] + \frac{g^2}{2} \nabla_x^2 p(x,t) \;, \qquad (3.90)
\end{aligned}$$

where we define

$$f(x,t) = \int dx_0 dx_1 p^{\text{train}}(x_0,x_1) \frac{f(x,t|x_0,x_1) p(x,t|x_0,x_1)}{p(x,t)} \;. \qquad (3.91)$$

With this drift function and a diffusion term $g(t)$, the solution of the FPE is given by Eq. (3.89). This gives us an SDE which propagates samples between $x_1 \sim p_1$ and $x_0 \sim p_0$, only depending on the current state $x(t)$. Starting from one of the distributions and numerically solving the SDE generates samples from the other distribution.

The last problem with the drift in Eq. (3.91) is that we cannot evaluate it analytically, so we encode it into a network f_θ. For this regression problem the natural loss is the mean squared error (MSE), but this requires training samples $f(x,t)$. We re-write this objective in terms of the conditional drift $f(x,t|x_0,x_1)$ defined in the SDE Eq. (3.76) and the conditional trajectories $p(x,t|x_0,x_1)$ defined in Eq. (3.95), as these allow for efficient generation of training samples. Following all steps from Section 3.4.2 the distribution mapping loss becomes

$$\begin{aligned}
\mathcal{L}_{\text{DM}} &= \left\langle \Big(f_\theta(x,t) - f(x,t|x_0,x_1) \Big)^2 \right\rangle_{t,(x_0,x_1)\sim p^{\text{train}}(x_0,x_1), x\sim p(x,t|x_0,x_1)} \\
&= \left\langle \left(f_\theta(x,t) - \frac{g(t)^2 (x-x_0)}{\bar{\sigma}(t)^2} \right)^2 \right\rangle_{t,(x_0,x_1)\sim p^{\text{train}}(x_0,x_1), x\sim p(x,t|x_0,x_1)} \;. \qquad (3.92)
\end{aligned}$$

The learned drift function depends on the pairing information in the training data, encoded via $p^{\text{train}}(x_0,x_1)$. Different pairings lead to different SDEs encoding different trajectories, but they all result in a generative network with the correct boundary distributions in Eq. (3.89).

Noise schedules for Schrödinger Bridges and Direct Diffusion

Choosing $g(t) = \sqrt{\beta(t)}$, with $\beta(t)$ the triangular function

$$\beta(t) = \begin{cases} \beta_0 + 2(\beta_1 - \beta_0)t & 0 \leq t < \dfrac{1}{2} \\ \beta_1 - 2(\beta_1 - \beta_0)\left(t - \dfrac{1}{2}\right) & \dfrac{1}{2} \leq t \leq 1 \end{cases} \tag{3.93}$$

and $\beta_0 = 10^{-5}$ and $\beta_1 = 10^{-4}$, we obtain the Schrödinger Bridge formulation [77, 81].

For constant $g(t) = g$, Eq. (3.75) simplifies to $\bar{\sigma}_{\text{ref}}(t) = g\sqrt{t}$ and $\sigma_{\text{ref}}(t) = g\sqrt{1-t}$. Consequently, Eq. (3.87) yields

$$\mu(t) = (1-t)x_0 + tx_1 \qquad \text{and} \qquad \sigma(t) = g\sqrt{t(1-t)}\;. \tag{3.94}$$

Our trajectory and probability distribution take the form

$$\begin{aligned} & x(t) = (1-t)x_0 + tx_1 + g\sqrt{t(1-t)}\,\epsilon \qquad \text{with } \epsilon \sim \mathcal{N}(0,1) \\ \Leftrightarrow \qquad & p(x(t), t|x_0, x_1) = \mathcal{N}\big(x(t)|(1-t)x_0 + tx_1, g\sqrt{t(1-t)}\big)\;. \end{aligned} \tag{3.95}$$

The noise term vanishes at the endpoints $t = 0, 1$ and takes its maximum at $t = 1/2$. This constructs an SDE that interpolates between two arbitrary distributions and reduces to what we call Direct Diffusion (DiDi) [2] for $g \to 0$. To see this we start from Eq. (3.76) and insert the training trajectory from Eq. (3.95),

$$\begin{aligned} dx(t) &= \frac{x(t) - x_0}{t}\,dt + g\,dW \\ &= \frac{(1-t)x_0 + tx_1 + g\sqrt{t(1-t)}\,\epsilon - x_0}{t}\,dt + g\,dW \\ &= (x_1 - x_0)\,dt + g\left[\frac{\sqrt{t(1-t)}\,\epsilon}{t}\,dt + dW\right]\;. \end{aligned} \tag{3.96}$$

For $g \to 0$ the training SDE reduces to the training ODE from DiDi, with the linear velocity field $x_1 - x_0$ and the distribution mapping loss reduces to the flow matching loss.

3.5.3 Conditional distribution mapping

In the last section we have constructed an SDE-based mapping between two arbitrary distributions. We now describe the new aspect of adapting this DM-formalism to reproduces the correct conditional distributions [82]. Specifically, we look at the trajectories $p(x, t|x_1)$ obtained when solving the learned SDE repeatedly from the same starting point $x_1 \sim p_1$, and modify our formalism such that for $t \to 0$ it reproduces the correct data pairing $p^{\text{train}}(x_0|x_1)$.

First, we check how this density looks in our conditional training trajectories $p(x(t), t|x_0, x_1)$ by marginalizing over x_0. Similar to Eq. (3.89) we can write

$$p(x, t|x_1) = \int dx_0\; p^{\text{train}}(x_0|x_1)\; p(x(t), t|x_0, x_1)$$

$$= \int dx_0 \, p^{\text{train}}(x_0|x_1) \, \mathcal{N}(x|\mu(t), \sigma(t)) = \begin{cases} p^{\text{train}}(x|x_1) & t = 0 \\ \delta(x_1 - x) & t = 1 \, . \end{cases} \tag{3.97}$$

This conditional density has the correct boundary behavior. When conditioned on a latent-space event $x_1 \sim p_{\text{latent}}$, the density converges to a delta peak around this event at time $t = 1$ and converges to the training pairing conditional distribution $p^{\text{train}}(x|x_1)$ at time $t = 0$.

However, there is no guarantee that the generative SDE defined via the drift from Eq. (3.91) shares these conditional densities. We have derived this drift f by showing that it solves the same unconditional FPE in Eq. (3.90) as our training process, so we are only sure that they share the unconditional density $p(x, t)$.

We need a drift function leading to an SDE that shares the conditional density $p(x, t|x_1)$ of the training trajectories. This can be achieved by going through the same derivation as we did for the drift initially, but this time with the FPE for the conditional density

$$\begin{aligned} \frac{\partial p(x, t|x_1)}{\partial t} &= \int dx_0 p^{\text{train}}(x_0|x_1) \frac{\partial p(x, t|x_0, x_1)}{\partial t} \\ &= \int dx_0 p^{\text{train}}(x_0|x_1) \left[-\nabla_x [f(x, t|x_0, x_1) p(x, t|x_0, x_1)] + \frac{g^2}{2} \nabla_x^2 p(x, t|x_0, x_1) \right] \\ &= -\nabla_x \left[p(x, t|x_1) \int dx_0 p^{\text{train}}(x_0|x_1) \frac{f(x, t|x_0, x_1) p(x, t|x_0, x_1)}{p(x, t|x_1)} \right] \\ &\quad + \frac{g^2}{2} \nabla_x^2 \int dx_0 p^{\text{train}}(x_0|x_1) p(x, t|x_0, x_1) \\ &= -\nabla_x [p(x, t|x_1) f(x, t|x_1)] + \frac{g^2}{2} \nabla_x^2 p(x, t|x_1) \, . \end{aligned} \tag{3.98}$$

Here, we identify the conditional drift term to be

$$f(x, t|x_1) = \int dx_0 \, p^{\text{train}}(x_0|x_1) \frac{f(x, t|x_0, x_1) p(x, t|x_0, x_1)}{p(x, t|x_1)} \, . \tag{3.99}$$

Sampling a latent $x_1 \sim p_{\text{latent}}$ and solving the SDE with this drift function samples from the conditional density $p^{\text{train}}(x|x_1)$. This new conditional drift is not only a function of the current state of the SDE, but also of the initial state x_1. This additional input acts as the condition under which we want to generate a sample from the data distribution, similar to the way the CFM velocity is a function of the current state of the ODE and of the condition.

Loss function

This conditional drift is once again encoded in a network. Repeating the derivation of Section 3.4.2, we obtain the conditional distribution mapping (CDM) loss

$$\begin{aligned} \mathcal{L}_{\text{CDM}} &= \left\langle \Big(f_\theta(x, t, x_1) - f(x, t|x_0, x_1) \Big)^2 \right\rangle_{t, (x_0, x_1) \sim p(x_0, x_1), x \sim p(x, t|x_0, x_1)} \\ &= \left\langle \Big(f_\theta(x, t, x_1) - \frac{g(t)^2 (x - x_0)}{\bar{\sigma}(t)} \Big)^2 \right\rangle_{t, (x_0, x_1) \sim p^{\text{train}}(x_0, x_1), x \sim p(x, t|x_0, x_1)} \, . \end{aligned} \tag{3.100}$$

It is identical to the standard DM-loss, except for the network also using the initial condition x_1 as a third input. This is the only change necessary to allow the network to learn the proper conditional densities.

CHAPTER **4**

Fast Event Generation

In high energy physics we are in the privileged position of having reliable numerical tools to simulate physical events with great precision as described in Section 2.2. However, as the data statistic is ever increasing, so is the need for precise simulations. Accurate simulations come with a great computational cost, motivating the study of fast event generators. In this chapter, we explore options to speed up LHC event generation with ML tools, such as those presented in Chapter 3. In Section 4.1, we start by studying two simple toy examples to investigate the fitting behavior of DDPMs, before moving on to End-to-End generation of Z+jets events. Motivated by the need for accurate off-shell simulations in $t\bar{t}$-decays, we explore options for fast event generation with full off-shell effects using DiDi in Section 4.2. Lastly, we investigate speeding up costly calorimeter simulations using CFMs with a vision transformer architecture [83] in Section 4.3.

4.1 End-to-End event generation

The content of this section was finalized in collaboration with Anja Butter, Nathan Huetsch, Tilman Plehn, Jonas Spinner and Peter Sorrenson. It is adapted from Ref. [1].

The goal of this study is to investigate DDPMs and evaluate whether their performance makes them suitable for LHC applications. As presented in Section 3.4.1 we develop uncertainty-aware B-DDPMs by bayesianizing the networks' architecture and loss function. We start with two simple toy models and then move on to the LHC phase space of Z+jets events.
All the networks presented in this section are implemented in PyTorch [84] and use Adam [60] as optimizer. All hyperparameters are given in Tab. A.1. We use a simple residual network which consists of multiple fully connected dense layers with SiLU activation functions [85]. Within our setup a significant increase in performance is achieved when initializing the weights of the last layer in each block to zero. In our setup dropout layers lead to significantly worse results, while normalization layers have no visible impact on the results. We find that the training of DDPM models can be very noisy, using a

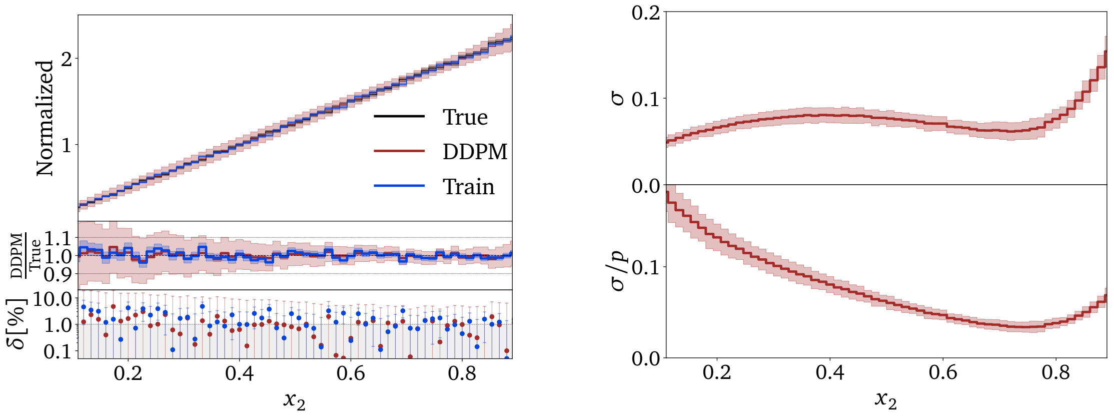


Figure 4.1: Ramp distribution from the DDPM. We show the learned density and its B-DDPM uncertainty (left) as well as the absolute and relative uncertainties with a range given by 10 independent trainings (right). We use $\delta = |\text{Model} - \text{Truth}|/\text{Truth}$.

large batch size can help to stabilize this. In general, training diffusion models requires a relatively large number of epochs, as indicated in Tabs. A.1. A key result of our study is to use a one-cycle scheduling for the DDPMs, as well as significantly downsizing the models compared to INNs, to allow for more training epochs.

4.1.1 Toy models and Bayesian networks

Before we can turn to the LHC phase space as an application to our novel generative model, we study its behavior for two simple toy models, directly comparable to Bayesian INNs [66]. These toy models serve two purposes: first, we learn about the strengths and the challenges of the network architecture, when the density estimation task is simple and the focus lies on precision. Second, the interplay between the estimation of the density and its uncertainty over phase space allows us to understand how the network encodes the density. We remind ourselves that an INN just works like a high-dimensional fit to the correlated 2-dimensional densities [66].

Our first toy example is a normalized ramp, linear in one direction and flat in the second,

$$p_{\text{ramp}}(x_1, x_2) = 2x_2 \ . \tag{4.1}$$

The network input and output are unweighted events. A training dataset of 600k events guarantees that for our setup and binning the statistical uncertainty on the phase space density is around the per-cent level. To show one-dimensional Bayesian network distributions we sample the x_i-direction and the θ-space in parallel [41, 66]. This way the uncertainty in one dimension is independent of the existence and size of other dimensions.

We show the non-trivial one-dimensional distributions in Fig. 4.1. In the left panel we see that the network learns the underlying phase space density well, but not quite at the desired per-cent precision. The uncertainty from the B-DDPM captures remaining deviations, if anything, conservatively. In the right panel we see that the absolute uncertainty has a minimum around $x_1 = 0.7$, similar to the behavior of the Bayesian

INN and confirmed by independent trainings. We can understand this pattern by looking at a constrained fit of the normalized density

$$p(x_2) = ax_2 + b = a\left(x_2 - \frac{1}{2}\right) + 1 \qquad \text{with} \qquad x_2 \in [0,1] \ . \tag{4.2}$$

A fit of a then leads to an uncertainty in the density of

$$\sigma \equiv \Delta p \approx \left| x_2 - \frac{1}{2} \right| \ \Delta a \ , \tag{4.3}$$

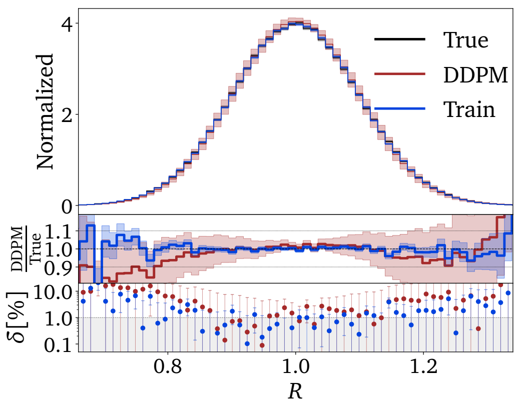
just using simple error propagation. The minimum in the center of the phase space plan can be interpreted as the optimal use of correlations in all directions to determine the local density.

For the DDPM the minimum is not quite at $x_2 = 0.5$, and the uncertainty as a function of x_2 is relatively flat over the entire range. Because of the statistically limited training sample, the network output comes with a relatively large uncertainty towards $x_2 = 0$. For larger x_2-values, the gain in precision and uncertainty is moderate. For $x_2 > 0.75$ the absolute and relative uncertainties increase, reflecting the challenge to learn the edge at $x_2 = 1$. These results are qualitatively similar, but quantitatively different from the INN case, which benefits more from the increase in training data and correlations for $x_2 = 0.1$ ... 0.5.

The second toy example is a Gaussian ring, or a Gaussian sphere in two dimensions,

$$p_{\text{ring}}(x_1, x_2) = \mathcal{N}(\sqrt{x_1^2 + x_2^2}; 1, 0.1). \tag{4.4}$$

The DDPM result are shown in Fig. 4.2. The precision on the density is significantly worse than for the ramp, clearly missing the per-cent mark. The agreement between the training data and the learned density is not quite symmetric, reflecting the fact that we train and evaluate the network in Cartesian coordinates but show the result in R. Especially for large radii, the network significantly overestimates the tail, a failure mode which is covered by the predictive uncertainty only for $R \lesssim 1.3$. In the right panels of Fig. 4.2 the main feature is a distinct minimum in the uncertainty around the mean of the Gaussian. As for the ramp, this can be understood from error propagation in a



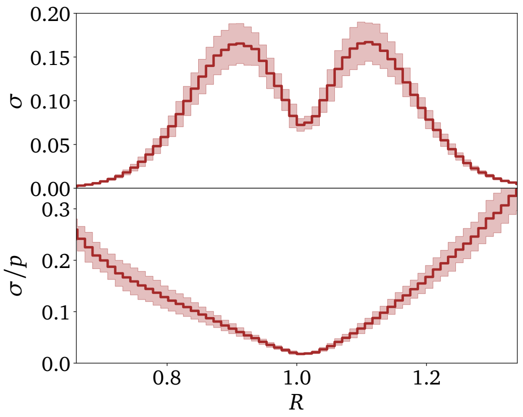


Figure 4.2: Gaussian ring distribution from the DDPM. We show the learned density and its B-DDPM uncertainty (left) as well as the absolute and relative uncertainties with a range given by 10 independent trainings (right).

constrained fit. If we assume that the network first determines a family of functions describing the radial dependence, in terms of a mean and a width, the contribution from the mean vanishes at $R = 1$ [66]. Alternatively, we can understand the high confidence of the network through the availability of many radial and angular correlations in this phase space region. Summarizing our findings, DDPMs behave similar but not identical to the INN. The relation between the density and its uncertainty shows patterns of a constrained fit, suggesting that during the the training the networks first determine a class of suitable models and then adjust the main features of these models, like the slope of a ramp or the position and width of a Gaussian ring.

4.1.2 LHC events

Most generative network tasks at the LHC are related to learning and sampling phase space densities, for instance event generation at the parton or reconstruction level, the description of detector effects at the reconstruction level, the computation of event-wise likelihoods in the matrix element method, or the inversion and unfolding of reconstructed events. This is why we benchmark our new networks on a sufficiently challenging set of LHC events. Following Ref. [41] we choose the the production of leptonically decaying Z-bosons, associated with a variable number of QCD jets,

$$pp \to Z_{\mu\mu} + \{1, 2, 3\} \text{ jets} . \tag{4.5}$$

The network has to learn the sharp Z-peak as well as correlated phase space boundaries and features in the jet-jet correlations. We generate the training dataset of 5.4M events (4.0M + 1.1M + 300k) using Sherpa2.2.10 [24] at 13 TeV, including ISR and parton shower with CKKW merging [86], hadronization, but no pile-up. The jets are defined by Fastjet3.3.4 [53] using the anti-k_T algorithm [50] and applying the basic cuts

$$p_{T,j} > 20 \text{ GeV} \qquad \text{and} \qquad \Delta R_{jj} > 0.4 . \tag{4.6}$$

The jets and muons are each ordered in transverse momentum. Our phase space dimensionality is three per muon and four per jet, i.e. 10, 14, and 18 dimensions. Momentum conservation is not guaranteed, because some final-state particles might escape for instance the jet algorithm. However, the physically relevant phase space dimensionality is reduced to 9, 13, and 17 by removing the global azimuthal angle.

Our data representation includes a minimal preprocessing. Each particle is represented by

$$\{ \, p_T, \eta, \phi, m \, \} . \tag{4.7}$$

Given Eq.(4.6), we provide the form $\log(p_T - p_{T,\text{min}})$, leading to an approximately Gaussian shape. All azimuthal angles are given relative to the leading muon, and the transformation into $\text{artanh}(\Delta\phi/\pi)$ again leads to an approximate Gaussian. The jet mass is encoded as $\log m$. Finally, we centralize and normalize each phase space variable as $(q_i - \bar{q}_i)/\sigma(q_i)$ and apply a whitening/PCA transformation separately for each jet multiplicity for the two diffusion models.

The additional challenge for Z+jets event generation is the variable number of jets, which we tackle with a conditional evaluation [41], illustrated in Fig. 4.3. The training is independent for the three jet multiplicities. We start by giving the information for the $Z + 1$-jet sub-process, 12 phase space dimensions, to a first network. It is supplemented

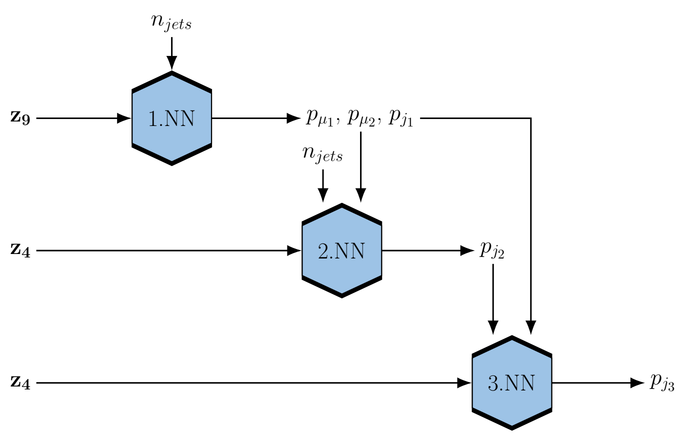


Figure 4.3: Conditional Sampling Architecture.

with the one-hot encoded jet count. The second network then receives the 4-momentum of the second jet as an input, and the $Z+1$-jet information additionally to the jet count as a condition. Analogously, the third network learns the third jet kinematics conditioned on the $Z+2$-jet information. For democratic jets this conditioning would be perfect, but since we order the jets in p_T it has to and does account for the fact that for higher jet multiplicities the interplay between partonic energy and jet combinatorics leads to differences in the spectra of the leading jets at a given multiplicity.

As discussed in Section 3.4.1 time is a crucial condition for the DDPM network, and we embed it into the full conditioning of the LHC setup as a high-dimensional latent vector linked by a linear layer. The amount of training data is different for the different jet multiplicities and corresponding networks. As shown in Tab. A.1, the first network uses the full 3.2M events, the second 850k events with at least two jets, and the third network 190k events with three jets. This hierarchy is motivated by the way the chain of conditional networks add information and also by the increasing cost of producing the corresponding training samples. We could balance the data during training, but for the B-DDPM model this leads to a slight performance drop. We compensate the lack of training data by increasing the number of epochs successively from 1000 to 10000.

Going from toy models to LHC events, we increase the number of blocks to two, which improves the performance. The reason is that we attach the condition to the input at the beginning of each block, so the second block reinforces the condition. Going to even more blocks will slightly improve the performance, but at the expense of the training time.

In Fig. 4.4 we show a set of kinematic distributions for different jet multiplicities, including the jet-inclusive scalar sum of the up to three $p_{T,j}$. These distributions can be compared directly to the Bayesian INN results in Fig. 11 of Ref. [41], serving as a precision baseline. Starting with the almost featureless p_T-distributions in the left panels, we see that for all three distributions the deviation from the truth, given by high-statistics training data, is similar for the actual training data and for the DDPM-generated events. The network really extracts all available information from the training data combined with its fit-like implicit bias. For sufficient training statistics, the precision on the phase space density as a function of p_T is below the per-cent level, easily on par with the INN baseline. For a given jet multiplicity this precision drops with increasing p_T and correspondingly decreasing training data, an effect that is correctly and conservatively modeled by the uncertainty estimate of the B-DDPM. Combining all n-jet samples into one observable is no problem for the network and does not lead to any artifacts.

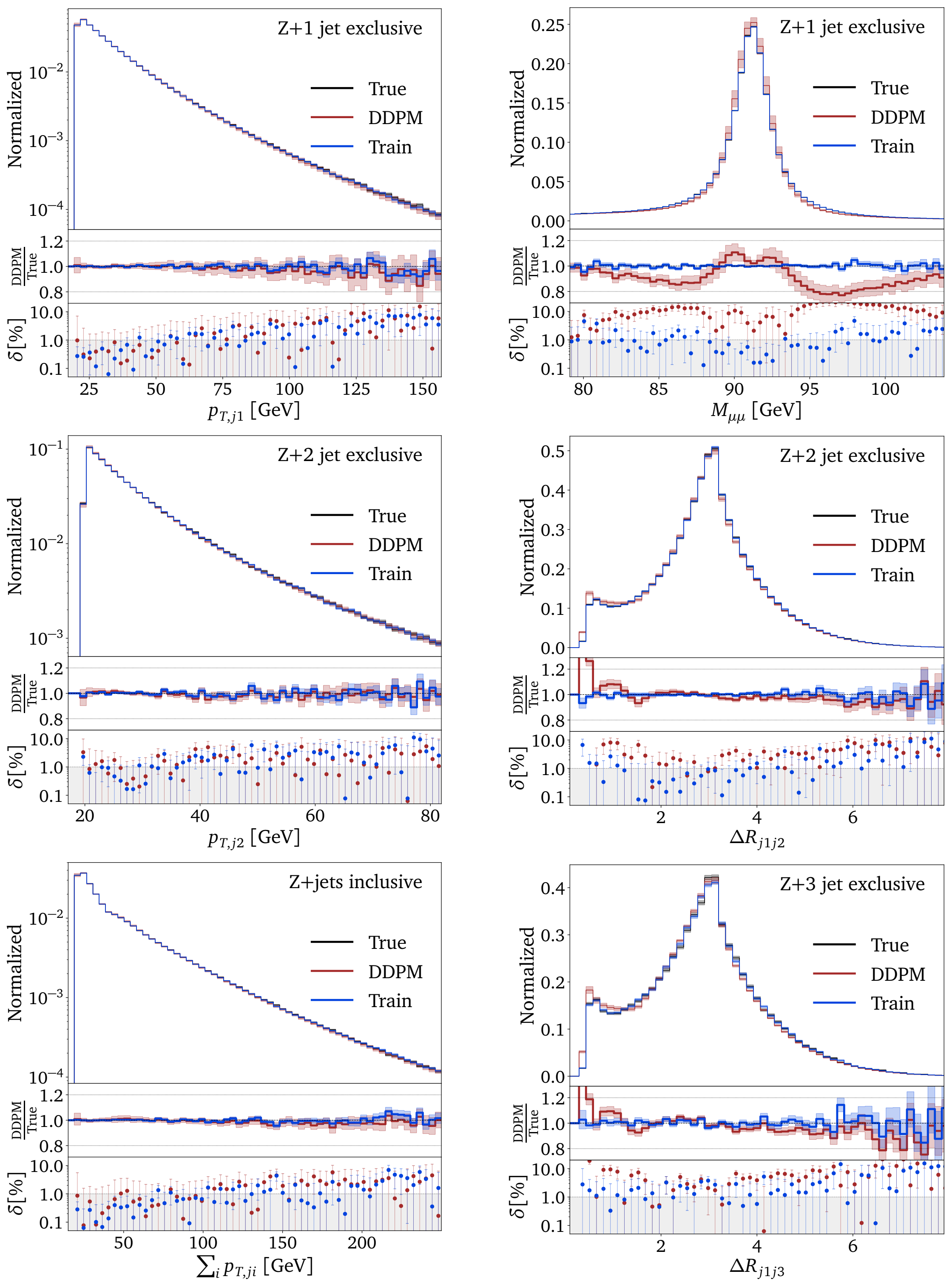


Figure 4.4: Bayesian DDPM densities and uncertainties for $Z+1$ jet (upper), $Z+2$ jets (center), and $Z+3$ jets (lower) from combined $Z+$ jets generation. The uncertainty on the training data is given by bin-wise Poisson statistics. The network architecture is given in Tab. A.1. For a comparison with the INN we refer to Fig. 11 of Ref. [41].

In the right panels of Fig. 4.4 we show the most challenging phase space correlations. We start with the Z-peak, which governs most of the events, but requires the network to learn a very specific phase space direction very precisely. Here, the agreement between the true density and the DDPM result drops to around 10% without any additional phase space mapping, similar to the best available INN. The deviation is not covered by the Bayesian network uncertainty, because it arises from a systematic failure of the network in the phase space resolution, induced by the network architecture. However, this effect is less dramatic than it initially looks when we notice that the ratio of densities just describes the width of the mass peak being broadened by around 10%. If needed, it can be easily corrected by an event reweighting of the Z-kinematics. Alternatively, we can change the phase space parametrization to include intermediate particles, but most likely at the expense of other observables.

Next, we study the leading challenge of ML-event generators, the jet-jet correlations and specifically the collinear enhancement right at the hard jet-separation cut of $\Delta R_{jj} > 0.4$. Three aspects make this correlation hard to learn: (i) this phase space region is a sub-leading feature next to the bulk of the distribution around $\Delta R_{jj} \sim \pi$; (ii) it includes a sharp phase space boundary, which density estimators will naturally wash out; and (iii), the collinear enhancement needs to be described correctly, even though it appears right at the phase space boundary. Finally, for this correlation the conditional setup and the Bayesian extension are definitely not helpful.

What helps for this correlation is the so-called magic transformation introduced in Ref. [41]. It scales the ΔR_{jj}-direction in phase space such that the density in this phase space direction becomes a monotonous function. While from a classic Monte Carlo perspective the benefits of this transformation are counter-intuitive, from a a fit-like perspective the magic transformation can simplify the class of function which the network then adapts to the data, as shown for the toy models in the previous section. This argument is confirmed by the fact that for our diffusion networks this transformation is helpful, just like for the INN. Both, for the 2-jet and the 3-jet sample we see that with the magic transformation the DDPM learns the ΔR_{jj} features, but at the same 10% level as the INN and hence missing our 1% target. The Bayesian uncertainty estimate increases in this phase space region as well, but it is not as conservative as for instance in the p_T-tails.

The challenge of current diffusion networks, also the DDPM, is the evaluation speed. For each additional jet we need to call our network 1000 times, so sampling 3-jet events takes three times as long as sampling 1-jet events. However, none of the networks presented in this study are tuned for generation speed, the only requirement for a limited hyperparameter scan is the precision baseline given by the INN.

Summarizing our findings, we have provided the first comprehensive study of strengths and weaknesses of DDPMs for an established LHC task. We have first implemented a Bayesian network version, which allows us to understand the different ways the B-DDPM approaches the density estimation. We have applied B-DDPMs to the generation of Z+jets events, with a focus on the conditional setup for variable jet multiplicities and the precision in the density estimation [41]. The most challenging phase space correlations are the narrow Z-peak and the angular jet–jet separation combined with a collinear enhancement.

DDPMs are, conceptually, not very far from the INNs. We found that they face the same difficulties, especially in describing the collinear jet–jet correlation. Just like for the INN, the so-called magic transformation [41] solved this problem. DDPM provided an excellent balance between expressivity and precision, at least on part with advanced INNs. This included the density estimation as well as the uncertainty map over phase space. The DDPM model is based on a proper likelihood loss, with all its conceptual and practical advantages for instance when bayesianizing it. It required long training, but fewer network parameters than then INN. We emphasize that ML-research on diffusion models is far from done, so all differences between the two models found in this study should be considered with a grain of salt.

Altogether, we found that new generative network architectures have the potential to outperform even advanced normalizing flows and INNs. However, diffusion models come with their distinct set of advantages and challenges. Given the result of our study we expect significant progress in generative network applications for the LHC, whenever the LHC requirements in precision, expressivity, and speed can be matched by a DDPM.

4.2 Generating off-shell effects in leptonic $t\bar{t}$-decays

The content of this section was finalized in collaboration with Anja Butter, Tomáš Ježo, Michael Klasen, Mathias Kuschick and Tilman Plehn. It is adapted from Ref. [2].

Fast and precise theoretical predictions from first principles are required for essentially every experimental LHC analysis. This is especially true for modern inference methods, where the complete phase space coverage ensures an optimal measurement [35, 87]. Based on perturbative quantum field theory, precision simulations of the hard scattering process face two challenges, the loop order and the multiplicity of the partonic final state [88]. The latter increases rapidly for example, when we describe the production of decaying heavy particles. Naively, one could expect that describing the decay kinematics close to the mass shell of the heavy particles, for example using a Breit-Wigner propagator, is sufficient. However, given the precision targets of the upcoming LHC runs, on-shell approximations are no longer justified [89, 90]. In view of the HL-LHC, simulations need to describe heavy particle production and decay including, both, quantum corrections and off-shell kinematics.
The goal of this study is to develop a precise and fast generative network that maps two LHC phase spaces onto each other [76, 77, 91]. If one phase space is a narrow sub-manifold of the other one, like in the case of on-shell and off-shell phase spaces of production of heavy unstable particles, one cannot use regression or classifier reweighting but must resort to generative networks.

In this study we present such a mapping and use it to efficiently generate events over an off-shell phase space from given on-shell events. The idea behind this sampling from on-shell events is that the generative network does not have to reproduce the on-shell features and can focus on the additional and relatively smooth off-shell extension. We start by describing our training dataset and the problem of off-shell event generation in Section 4.2.1. Our generative network setup, based on a Bayesian- Conditional Flow Matching (CFM) architecture, is presented in Section 4.2.2. To control the generative network performance [92] and to improve the precision of the kinematic distributions [41], we apply a classifier reweighting in Section 4.2.3.

4.2.1 Off-shell vs. on-shell events

Motivated in Section 2.3, our benchmark process is the complete off-shell top pair production followed by leptonic decays,

$$pp \to b e^+ \nu_e \, \bar{b} \mu^- \nu_\mu \, . \tag{4.8}$$

Top pair production with its rich resonance structure is known to challenge phase space sampling, just as generative networks for transition amplitudes [93].

In the factorized approach, in which the production and decay processes decouple, this process is known all the way up to NNLO-QCD [94–97], NLO-EW [98–102] and NNLO-QCD combined with NLO-EW [103] in the production process; NNLO-QCD [104, 105] in the decay; and NNLO-QCD in both production and decay [106–108]. Full off-shell calculations in the dilepton channel are so far only available at NLO-QCD [109–114], but a calculation of full off-shell top pair production in association with an extra jet at NLO is also available in Ref. [115]. In this pilot study we restrict ourselves to leading order in QCD, $\mathcal{O}(\alpha_S^2 \alpha^4)$, and reserve the application of our method to higher-order predictions to a future publication.

We generate event samples for 13 TeV proton-proton collisions with NNPDF31_nlo_as_0118 parton distributions [116]. The neutrinos, charged leptons and quarks of the first two generations are treated as massless, and the CKM matrix is assumed to be trivial. All input parameters are given in Tab. 4.1, and the electromagnetic coupling α and the weak mixing angle are derived from the weak gauge-boson masses and the Fermi constant.

Our study and our results can be extended to higher-order predictions or other processes by combining different jet multiplicities in the final state [37]. Our two benchmark datasets are generated with Hvq [117] and Bb4l [118, 119], respectively. The data generated with Hvq only includes approximate off-shell effects using a finite top width and including spin correlations [120] and is referred to as on-shell data. The generator is based on the Powheg method [121, 122] and is part of the Powheg Box V2 [123] package.

The data generated with Bb4l takes into account full off-shell effects also including singly-resonant and non-resonant contributions and the corresponding interferences. The generator employs the PowhegRes method [124], tailored for simulations with unstable particles. In this case, W bosons and b quarks do not always stem from a top decay.

m_t	172.5 GeV	Γ_t	1.453 GeV
m_b	4.75 GeV		
m_Z	91.188 GeV	Γ_Z	2.441 GeV
m_W	80.419 GeV	Γ_W	2.048 GeV
m_H	125.0 GeV	Γ_H	0.0403 GeV
$\mathcal{B}(W \to e\nu/\mu\nu)$		1/9	
G_F		$1.16585 \times 10^{-5}\text{GeV}^{-2}$	

Table 4.1: Parameters used for the generation of the training datasets.

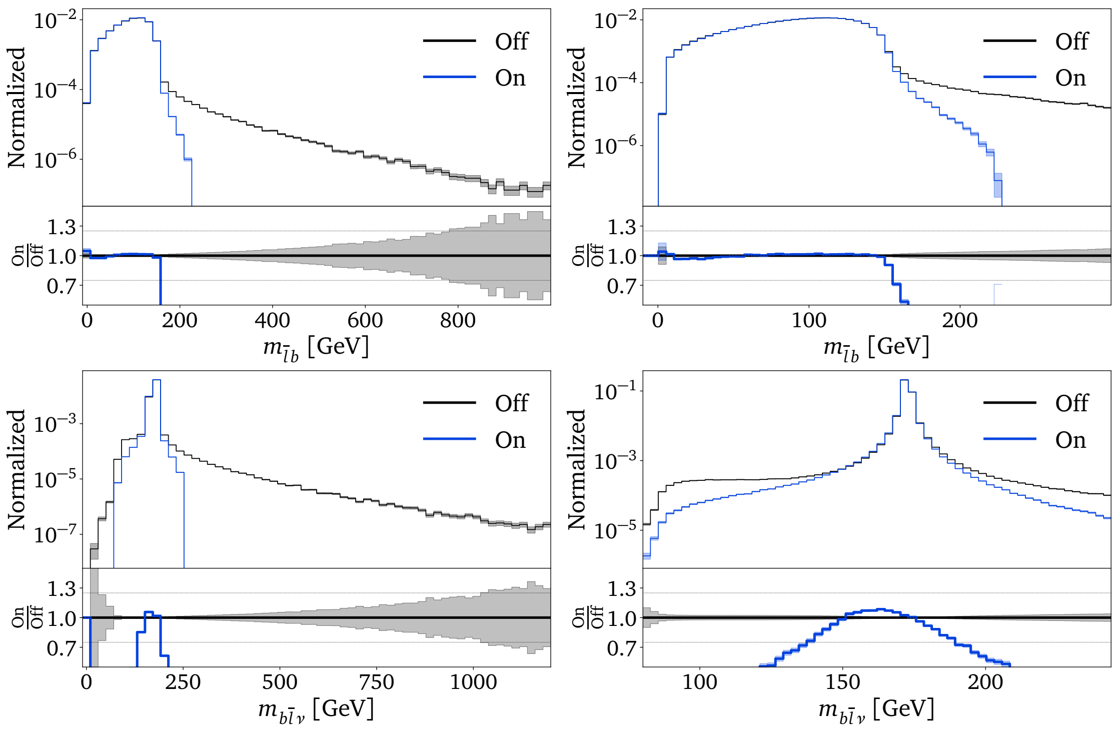


Figure 4.5: Example distributions for on-shell and off-shell $t\bar{t}$ event samples, illustrating the different phase space coverage. The right panels show the same distribution as the left panels, but zoomed into the respective bulk region.

In Fig. 4.5 we illustrate the size of the off-shell effects for a selection of kinematic distributions of final-state leptons and b-quarks. These distributions are not meant to compare realistic predictions with different treatment of off-shell effects, but rather our two datasets, so no event selection criteria are applied. For the invariant mass of the lepton-b system we ensure, through charge identification, that the two particles come from the same (anti)top decay. Correspondingly, their invariant mass has an upper edge that does not exist for off-shell events. For the reconstructed top mass, or the invariant mass of the three decay products, we clearly see the Breit-Wigner propagator form, with an explicit cutoff. Far below the actual top mass, the off-shell prediction develops a shoulder at the W-mass. The secondary panels show the ratios of the integrated one-dimensional phase space densities, illustrating that a reweighting strategy between the two samples is unlikely to work.

The strategy behind our surrogate network is to generate an off-shell event dataset and learn its structures relative to a corresponding on-shell event dataset. This strategy can be applied to any process and at any order in perturbation theory. Our two datasets consist of 5M unit-weight events each. The six particles in the final state of Eq.(4.8) are represented by $\{p_T, \eta, \phi\}$, with fixed external particle masses. We remove three degrees of freedom through a global azimuthal rotation and by enforcing two-dimensional transverse momentum conservation, leaving us with a 15-dimensional phase space. Each on-shell requirement replaces a full phase space dimension by a fixed range, given by the Breit-Wigner shape with a hard-coded cutoff in the reconstructed invariant mass. Moreover, we conceal information relevant for eventual parton showering like the colourflow configuration or resonance history assignment. This is appropriate because the Bb4l generator in its default setup does not distinguish between the $t\bar{t}$ and single-top resonance

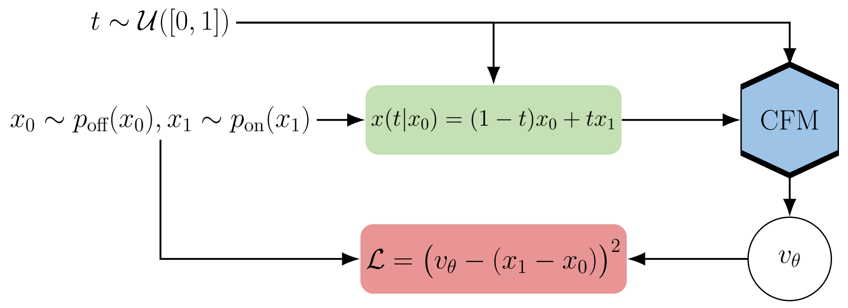

$t \sim \mathcal{U}([0,1])$
$x_0 \sim p_{\text{off}}(x_0), x_1 \sim p_{\text{on}}(x_1)$
$x(t|x_0) = (1-t)x_0 + tx_1$
CFM
v_θ
$\mathcal{L} = \left(v_\theta - (x_1 - x_0)\right)^2$

Figure 4.6: Training procedure for a CFM network mapping between on-shell and off-shell phase space distributions. Diagram adapted from Ref [1].

histories and assigns colourflow correspondingly.

For the network input, the kinematic variables are preprocessed: we scale the transverse momenta to $p_T^{1/3}$ and express the azimuthal angles as $\operatorname{arctanh}(\phi/\pi)$. As subsequent classifier input, it turns out that $p_T^{-1/3}$ leads to the best results. Finally, all input dimensions are standardized to zero mean and unit variance.

4.2.2 Direct Diffusion

For the network training, we start with the condition that we do not want to train on paired on-shell and off-shell events, because such a pairing does not follow a well-defined algorithm. This does not mean it is impossible to construct such a mapping, but we expect it to lead to artifacts. Instead, we will train a generative Direct Diffusion (DiDi) network on two phase space densities, one from on-shell and one from off-shell events. Following the formalism described in Section 3.5, we train our DiDi network by the usual CFM loss

$$\mathcal{L}_{\text{CFM}} = \left\langle [v_\theta((1-t)x_0 + tx_1, t) - (x_1 - x_0)]^2 \right\rangle_{t\sim\mathcal{U}([0,1]), x_0\sim p_{\text{off}}, x_1\sim p_{\text{on}}} . \tag{4.9}$$

Because the CFM setup does not include a likelihood, we can go directly from p_{on} to p_{off}, without extra effort due to detours like in Flows4Flows [91], and without any pairing between $x_0 \sim p_{\text{off}}$ and $x_1 \sim p_{\text{on}}$.

As usual, we use a Bayesian version of the generative network [41, 66] with the loss of Eq. 3.57, to extract uncertainties on the learned phase space density. It includes a hyperparameter c to balance the regular CFM loss with the Bayesian regularization term. If the first loss term was a likelihood loss, this factor would be fixed by Bayes' theorem. We have checked that the network performance is stable over many orders of magnitudes for $c = 10^{-10}$ ... 10^{-2}. For larger values we observe that the training becomes unstable, as expected, while for very small values the uncertainty can no longer be captured. In addition, the prior weight distribution $q(\theta)$ is given by a unit Gaussian, where the choice of width hardly affects the network performance. Compared to the deterministic counterpart, this Bayesian network is equally precise. The network and training setup is visualized in Fig. 4.6. We use a simple dense network with SiLU activation [85], where the last layer is initialized at zero. This sets the initial velocity field to zero and induces an identity mapping at the starting point of the training. We do not

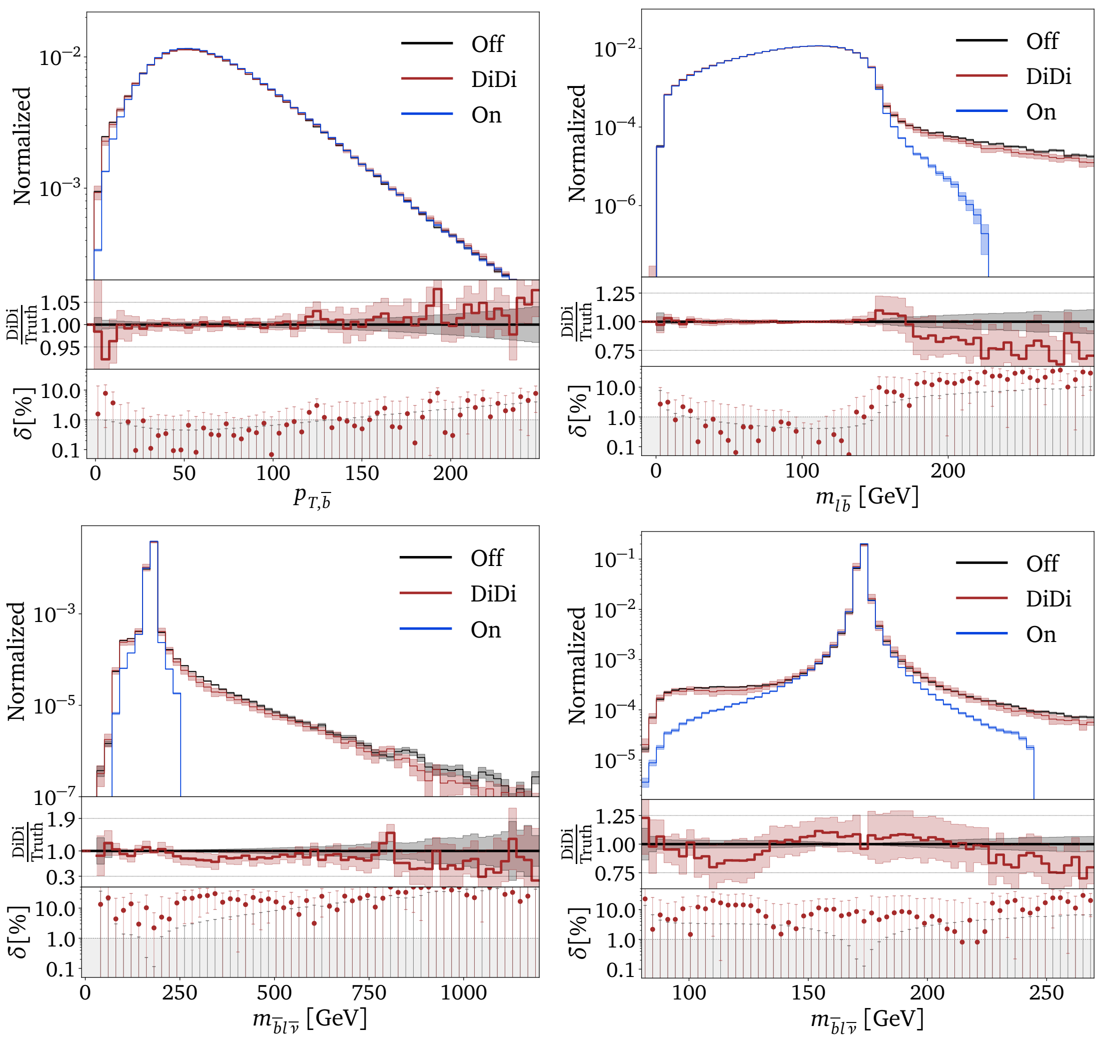


Figure 4.7: Results from our Direct Diffusion off-shell generator, compared to the on-shell starting point and the off-shell training distributions.

enforce a prescription for turning a given on-shell event into an off-shell event through the training data or the training procedure. In fact, for each epoch different phase space points will be connected via a linear trajectory. Instead, the network training constructs its own mapping to populate the off-shell phase space. This happens as part of the loss minimization, which means the transport map follows from an implicit measure encoded in the network architecture and loss.

As the CFM training objective is a simple regression, we can train on 17 dimensions, including two redundant degrees of freedom, as this happens to increase the precision. The two additional observables, the transverse momentum and the polar angle of the neutrino, are determined by transverse momentum conservation and are hence multidimensional correlations. Empirically, we found that it is easier for the model to learn the behavior of those observables when handed directly. Especially more complicated correlations such as the invariant mass of the reconstructed top benefit greatly from this additional information. While improving the efficiency of the training these dimensions will be ignored for the actual event generation.

The network hyperparameters and the training parameters are given in Tab. A.2. We

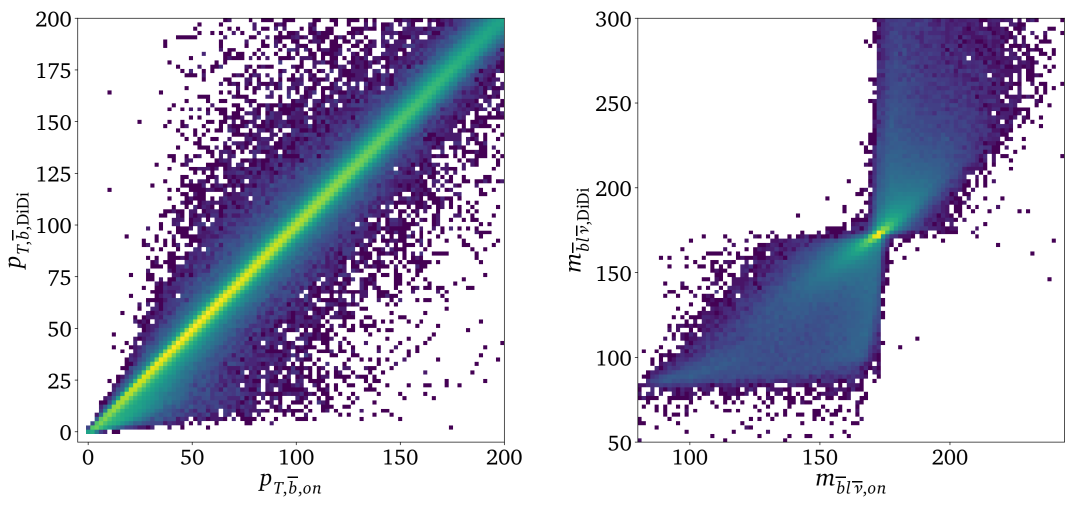


Figure 4.8: Migration plot — correlation between generated (off-shell) and starting (on-shell) distributions for two kinematic correlations. It illustrates the mapping found by the network during training.

encode t in a higher embedding dimension and following [40] we add batch-wise random noise of scale 10^{-4} to x_0 and x_1 during training. We use the standard *Dopri5* ODE solver to sample from our network. In the interest of precision we use a large batch size. This is a problem for generic optimal-transport networks [125], but can be easily implemented in our architecture. We tested the OT-CFM [125] and found that in our setup the required small batch size indeed led to worse performance.

In Fig. 4.7 we show a set of one-dimensional kinematic distributions, for the on-shell data we start from, the off-shell training data, and the generated off-shell data. In the first panel we see how the network learns subtle differences almost perfectly well. The typical precision is around the per-cent level. For complex and sensitive correlations, like the lepton-b invariant mass and the reconstructed top mass we start with a huge deviation between the on-shell distribution and the off-shell target. To generate these distributions correctly, the generative network has to learn correlations in the corresponding 9-dimensional sub phase space.

Our results confirm that in phase space regions covered well by both datasets, the generative network reproduces the target distribution precisely, as one could expect. However, we also see that even in phase space regions not populated by on-shell events the target distribution is reproduced relatively precisely. The typical agreement between the generated and target densities is around 10% in phase space regions with relatively little training data. While this deviation is covered by the uncertainties of the Bayesian network, we propose possible improvements in the next section.

Finally, we can ask how our generative network fills the off-shell phase space from the on-shell events. In Fig. 5.6 we show the correlations between the generated off-shell distribution and the on-shell starting distributions, i.e. the migration of paired latent and target phase space events from the forward simulation, where we emphasize that the pairing is only defined by the network evaluation, not by the training. In general, the correlation between the kinematic observables should be close to the identity, as confirmed

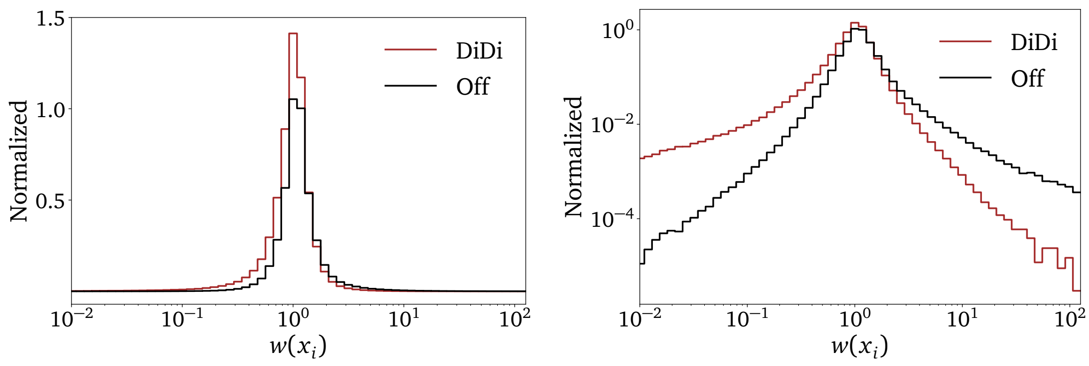


Figure 4.9: Histogram of the learned event weights, evaluated on the off-shell training data and the DiDi-generated off-shell events. The two panels show the same weights, on linear and logarithmic scales.

by the $p_{T,\bar{b}}$ distribution in the left panel. This means that the shift from on-shell to off-shell phase space is relatively small and uncorrelated. However, for the reconstructed top mass, some of the events have to be shifted by a larger value, as illustrated in the right panel. While the width of the linear correlation becomes very small around the top mass peak, it rapidly increases away from the peak, demonstrating the large shift required to populate the off-shell phase space. Moreover, our network maps events from each side of the Breit-Wigner peak to the same side of the off-shell distribution.

4.2.3 Classifier control and reweighting

Because the unsupervised density estimation underlying our generative network is more challenging and less precise than a supervised classifier training, we can use a trained classifier as a function of phase space to systematically check and improve the generative network. As discussed in Section 3.2, a perfectly trained statistical classifier converges to the ratio of likelihoods,

$$C(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)} \,. \tag{4.10}$$

As a function of phase space we can use this classifier to construct an event reweighting, which improves the precision of the generative networks [41],

$$w(x) = \frac{p_{\text{data}}(x)}{p_{\text{model}}(x)} = \frac{C(x)}{1 - C(x)} \,. \tag{4.11}$$

In addition, we can use the same learned event weights to determine the precision of the generative network and systematically search for failure modes by searching for clusters of very small or very large weights in phase space [92].

We train the classifier on 27 observables, our 15 physical dimensions, complemented by the reconstructed top and anti-top masses, the reconstructed W^+ and W^- masses, the reconstructed masses of the $\bar{b}l$ and $b\bar{l}$ systems, and the six corresponding transverse momenta. For the input of our classifier we sample events from our Bayesian generator setting each network weight to its mean value. The setup is given in Tab. A.3.

We show histograms of learned phase space weights $w(x_i)$ on a linear and a logarithmic

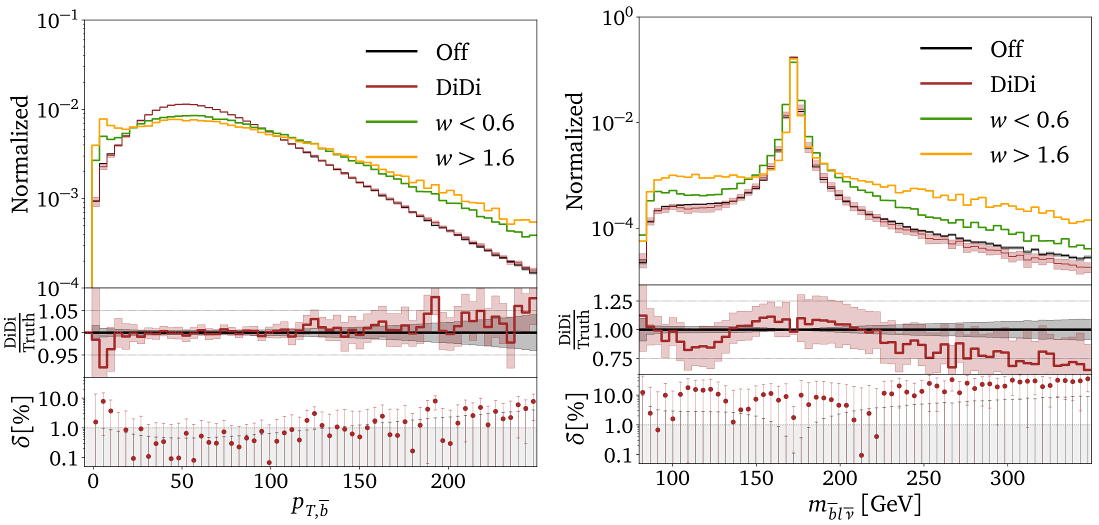


Figure 4.10: Clustering of the classifier trained to distinguish the off-shell training data from DiDi-generated events, for two example distributions.

axis in Fig. 4.9. As expected, the distributions peak at unit weights, with a width around 0.3. Following Eq.(4.11), large weights correspond to phase space regions where the generative network produces a too small density of off-shell points; small weights mark phase space regions where the generative network produces too many off-shell events, compared to the training data. To study both tails of the weight distribution, we evaluate the weights over the combination of 2M training and 2M generated events and confirm that both tails decrease rapidly. We eventually clip the event weights to $w < 15$ to improve the numerical behavior of our generation and avoid sparks in regions of low statistics.

In Fig. 4.10 we track phase space regions with very small and very large weights. We compare all events of our test sample to the subsets with $w(x) < 0.6$, corresponding to 12.3% of the sample, and $w(x) > 1.6$, corresponding to 6.2% of the generated sample. The two shown distributions illustrate the general feature that the kinematic distributions for both tails are similar. In the $p_{T,\bar{b}}$-distribution we can identify two limitations of the network training: for small transverse momenta a slight shift of the cliff will lead to large relative weight corrections, while for large transverse momenta the decreasing density of training events will increase the relative size of the noise. For the reconstructed anti-top mass, one of the critical distributions for our generative task, the low-weight and high-weight tails are also comparable with each other and also comparable on both sides of the mass peak. The only distinct feature appears around the peak where events with small weight dominate the slightly over populated sides of the peak. This indicates that all weight tails are generated by noise, there are no missing localized features, and the limiting factor is the statistics of the training data.

Even though the shortcomings of our generative networks, visible in Fig. 4.7, arise from noisy network training and do not reflect systematic shortcomings, they affect the trained generative network in a systematic, localized manner. As a function of phase space, we can correct them using the event weights from Eq.(4.11), because the classifier network is more sensitive and more precise than the generator network [92]. In Fig. 4.11 we show a set of kinematic distributions for reweighted events, where the uncertainty is given by the Bayesian generator. Comparing the agreement between the reweighted and

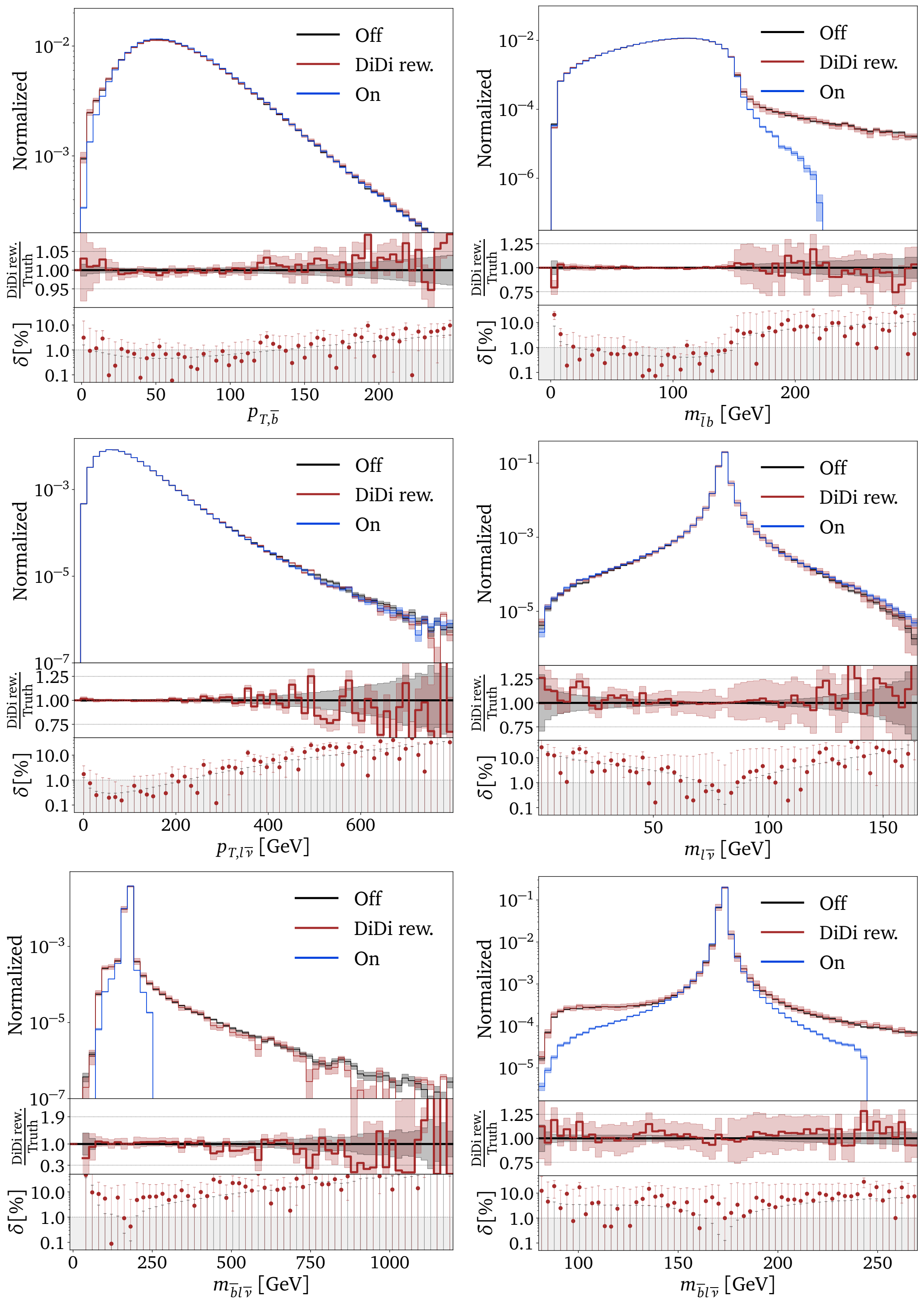


Figure 4.11: Results from our re-weighted Direct Diffusion off-shell generator, compared to the on-shell starting point and the off-shell target distributions.

the target distribution with the unweighted performance from Fig. 4.7 we see significant improvements. In the secondary panels we show the reweighted predictions from the Bayesian generator, allowing us to compare the statistical uncertainty from the training data with the predictive uncertainty from the generative network. For the entire range of $p_{T,\bar{b}}$ the network agrees with the true distribution within its predictive uncertainties, and almost within the uncertainties of the training data. For $m_{\bar{l}b}$, DiDi has to cover phase space far beyond the on-shell structures, and, again it hardly exceeds the statistical uncertainties of the training data and provides a conservative uncertainty estimate from the Bayesian setup.

The momentum and the mass of the reconstructed W^- match the truth perfectly in the bulk, and roughly within the statistics of the training data in the tails. The former is important, because it confirms the original motivation that the network reproduces the on-shell features with extremely high precision. The same can be seen for the reconstructed anti-top mass, where most of the phase space is filled by extrapolating from the on-shell distribution and the network learns these new phase space regions without degrading the precision in the bulk at all.

To conclude, we proposed a new method, namely to generate off-shell configurations relative to given on-shell configurations. Its advantage is that the generative network only learns a controlled deviation from a simple unit transformation. This simplified task allows us to generate a rich resonance structure without the usual challenges in network size and precision training. Our approach can be extended to higher-orders as long as the corresponding event samples can be provided as training data. Such an extension is not trivial because with each extra particle in the final state, e.g. due to an extra emission in the NLO real correction, the dimension of the phase space increases. We do not foresee any conceptual issues however, as increasing the dimension of the phase space in our setup is straightforward and relatively small event samples were needed at LO, and both diffusion networks as well as Classifiers have been shown to scale well with dimension. The size of the off-shell effect, or more specifically the ratio of the full and approximate off-shell prediction for a given observable, could in principle change dramatically when changing orders. This is not a problem though, as DiDi is able to reproduce full off-shell predictions also in regions where the approximate off-shell prediction is vanishing.

4.3 Fast Calorimeter Simulations

The content of this section was finalized in collaboration with Luigi Favaro, Ayodore Ore and Tilman Plehn. It is adapted from Ref. [3].

In this study, we apply cutting-edge generative networks to calorimeter shower simulations. The high-dimensional phase spaces of calorimeter showers are a challenge to the established normalizing flows or INNs [126], and different variants of diffusion networks appear to be the better-suited architecture [127]. In addition to showing that these networks are able to simulate sparse phase space signals like calorimeter showers, we will explore which phase space dimensionalities we can describe with full-dimensional latent spaces and how a dimension-reduced latent representation affects the network performance.

Given the GEANT4 benchmark presented in Section 4.3, we will see that a factorized approach is most promising. In Section 4.3.1 we introduce a CFM network combined with an autoregressive transformer to learn the layer energies. Next, we combine it with a 3-dimensional vision transformer to learn the shower shapes. This combination can be trained on datasets 2 and 3 of the CaloChallenge to generate high-fidelity calorimeter showers. In this application the step from dataset 2 to dataset 3 motivates a switch from full-dimensional voxel representations to a dimension-reduced latent space [128]. In Section 4.3.2 we study, in some detail, how the full-dimension generative network encodes the calorimeter shower information for both datasets. To alleviate the computational challenges, we also show how a lower-dimensional latent representation helps us describe high-dimensional data like the Calo Challenge dataset 3. For quantitative benchmarking of the learned phase space distribution we employ a learned classifier test, indicating that the network precision for both datasets is at the per-cent level and the loss in precision from a reduced latent space is controlled, including its only failure mode, which are the sparsity distributions.

Data and preprocessing

To benchmark our new network architectures, we use dataset 2 (DS2) [129] and dataset 3 (DS3) [130] of the CaloChallenge 2022 [6]. Each set consists of 200k GEANT4 [27] electron showers: 100k for training/validation and 100k for testing. Showers are simulated over a log-uniform incident energy range

$$E_{\text{inc}} = 10^3 \text{ ... } 10^6 \text{ MeV} . \tag{4.12}$$

The physical detector has a cylindrical geometry with alternating layers of absorber and active material, altogether 90 layers. The voxelization following Ref. [131] combines an active layer and an absorber layer resulting in 45 concentric cylindrical layers.

The particle originating the shower always enters at the (0,0,0) location and defines the z-axis of the coordinate system. The number of readout cells per layer is defined in a polar coordinate system and it is different for DS2 and DS3. DS2 has a total of 6480 voxels: 144 voxels per layer, each divided into 16 angular and 9 radial bins. DS3 has a much higher granularity with 40500 total voxels, where the number of layers is unchanged but the angular and radial binning is 50×18. Both datasets have a threshold of 15.15 keV. While this is an unrealistic cut for practical applications, it provides a useful challenge to high-dimensional generative networks covering a wide energy range.

Preprocessing

We improve our training by including a series of preprocessing steps, similar to previous studies [126, 127, 132–134]. We split information on the deposited energy from its distribution over voxels by introducing energy ratios [135]

$$u_0 = \frac{\sum_i E_i}{f E_{\text{inc}}} \qquad \text{and} \qquad u_i = \frac{E_i}{\sum_{j \geq i} E_j} \quad i = 1, \dots, 44 , \tag{4.13}$$

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.(4.13), 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,

$$\begin{aligned} x_\alpha &= (1-2\alpha)x + \alpha \in [\alpha, 1-\alpha] \qquad \text{with} \quad \alpha = 10^{-6} \\ x' &= \log \frac{x_\alpha}{1-x_\alpha} \; . \end{aligned} \tag{4.14}$$

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.

4.3.1 CaloDREAM

In CaloDREAM[1], we employ two generative networks, one energy network and one shape network [135]. 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 the chain

$$\begin{aligned} u_i &\sim p_\phi(u_i|E_\text{inc}) \\ x &\sim p_\theta(x|E_\text{inc}, u) \; . \end{aligned} \tag{4.15}$$

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, as in Eq.(4.12).

Energy network — Transfusion

Both of our generative networks use the Conditional Flow Matching architecture described in Section 3.4.2.

For the energy network, we exploit the causal nature of the energy deposition in layers using an autoregressive transfusion architecture [37], as visualized in Fig 4.12. 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

[1]The code used for this study is publicly available at https://github.com/heidelberg-hepml/calo_dreamer

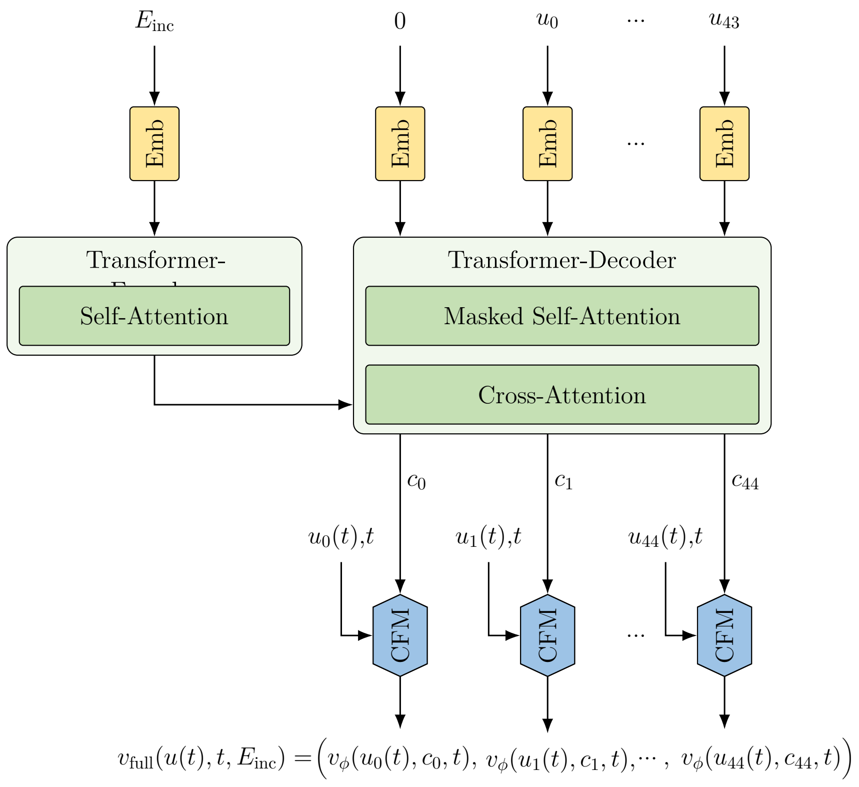


Figure 4.12: Schematic diagram of the autoregressive Transfusion network [37] used in our energy network.

and decoder outputs. The transformer outputs the vectors $c_0, \dots,\ c_{44}$, encoding the incident energy and previous energy ratios,

$$c_i = \begin{cases} c_i(u_0, \dots, u_{i-1}, E_{\text{inc}}) & i > 0 \\ c_i(E_{\text{inc}}) & i = 0\ . \end{cases} \tag{4.16}$$

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,

$$v_{\text{full}}(u(t), t, E_{\text{inc}}) = (v_\phi(u_0(t), c_0, t), \dots, v_\phi(u_{44}(t), c_{44}, t)) \tag{4.17}$$

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 Tab. A.4.

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 [83] and illustrated in Fig. 4.13. 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

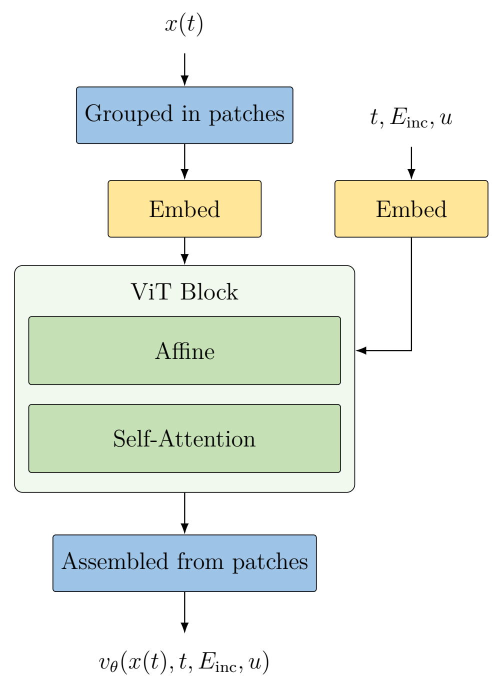


Figure 4.13: Schematic diagram of the vision transformer (ViT) [83] used in our shape network.

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

$$\begin{aligned} x_{\text{h}} &= x + \gamma_{\text{h}} g_{\text{h}}(a_{\text{h}} x + b_{\text{h}}), \\ x_{\text{l}} &= x_{\text{h}} + \gamma_{\text{l}} g_{\text{l}}(a_{\text{l}} x_{\text{h}} + b_{\text{l}}), \end{aligned} \tag{4.18}$$

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 Tab. A.5.

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. We decide on specific patch sizes for DS2 and DS3 through manual exploration.

Usually, we train Bayesian versions [66] of all our generative networks, including calorimeter showers [126]. 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.

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 [126, 128, 136].

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

$$\mathcal{L}_\text{VAE} = \big\langle -\log p_\psi(x|r) \big\rangle_{x\sim p_\text{data}, r\sim p_\psi(r|x)} + \beta \big\langle D_\text{KL}[p_\psi(r|x), \mathcal{N}(0,1)] \big\rangle_{x\sim p_\text{data}} . \tag{4.19}$$

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

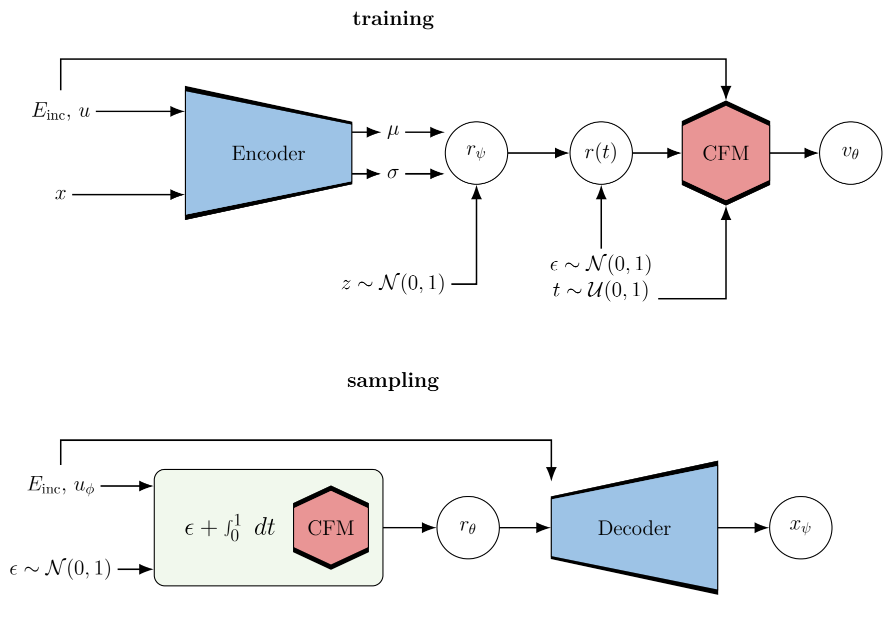


Figure 4.14: Training (upper) and sampling (lower) with the latent diffusion network, using a variational autoencoder.

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 do not expect the same reduction factor F to be optimal for both datasets.

The trained autoencoder is used as a pre- and postprocessing step for the CFM as illustrated in Fig. 4.14. Given the trained encoder distribution $p_\psi(r|x)$ the velocity field $v(r(t), t)$ imposes the boundary conditions

$$p(r,t) \to \begin{cases} \mathcal{N}(r;0,1) & t \to 0 \\ p_\psi(r|x) & t \to 1,\, x \sim p_{\text{data}}\,. \end{cases} \tag{4.20}$$

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 deterministic decoder $D_\psi(r)$. Here we summarize the sampling procedure, including the energy dependence, as three sequential steps:

$$\begin{aligned} u &\sim p_\phi(u|E_{\text{inc}}) \\ r &\sim p_\theta(r,1|u,E_{\text{inc}}) \\ x &= D_\psi(r,u,E_{\text{inc}}) \end{aligned} \tag{4.21}$$

All network hyperparameters and the main training parameters are given in App. A.

4.3.2 Results

Layer energies

In Fig. 4.15 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 on an independent test set. The comparison in terms of layer energy is shown in Fig. 4.16. The factorization procedure allows us to use the same

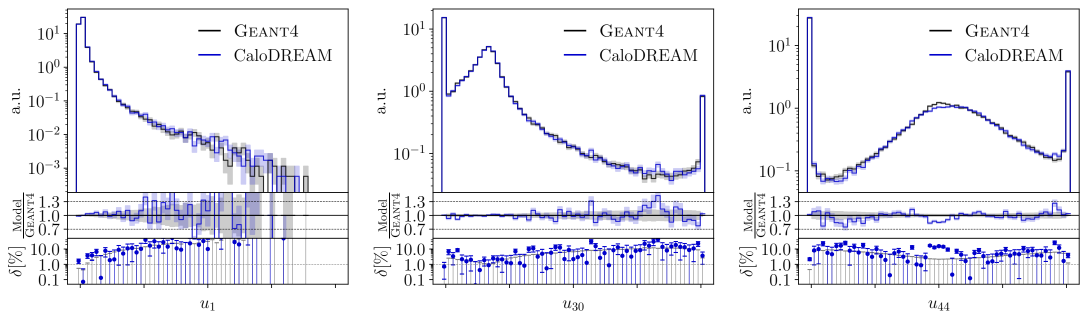


Figure 4.15: Distributions of selected u-features in DS2 from the CaloDREAM energy network (blue) compared to truth (grey). The error bars in all feature distributions in this section show the statistics of the respective datasets.

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 Section 4.3.1 to generate the actual calorimeter showers over the voxels. First, we evaluate the distribution of energy depositions per layer by looking at shape observables, like the center of energy of the shower and its width in the ϕ and η directions,

$$\begin{aligned}\langle\xi\rangle &= \frac{\xi\cdot x}{\sum_i x_i} \\ \sigma_{\langle\xi\rangle} &= \sqrt{\frac{\xi^2\cdot x}{\sum_i x_i} - \langle\xi\rangle^2} \qquad \text{for} \qquad \xi\in\{\eta,\phi\}\ .\end{aligned} \tag{4.22}$$

Here x_i is the energy deposition in a single voxel and the sum runs over the voxels in a layer.

In the first row of Fig. 4.16 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 data.

The second row of Fig. 4.16 shows example distributions probing the combination of layers. 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,

$$d_{r_j} = \frac{\sum_i^N k_i E_{i,r_j}}{E_{\text{tot},j}} \qquad r_j \in \{0,\dots,|r|\}\ . \tag{4.23}$$

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. This observable highlights a

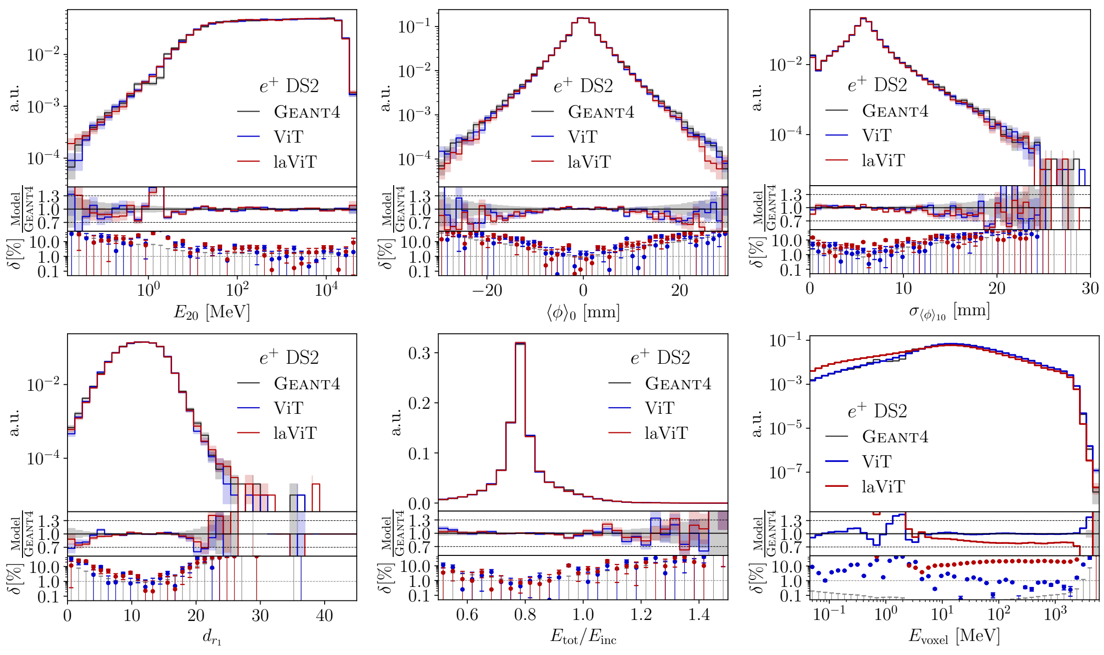


Figure 4.16: Selection of high-level features for DS2. The first row shows features for individual layers, the second row the combination of layers.

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. 4.16, we focus on the (latent) description with low-energy voxels. In Fig. 4.17 we again compare the two

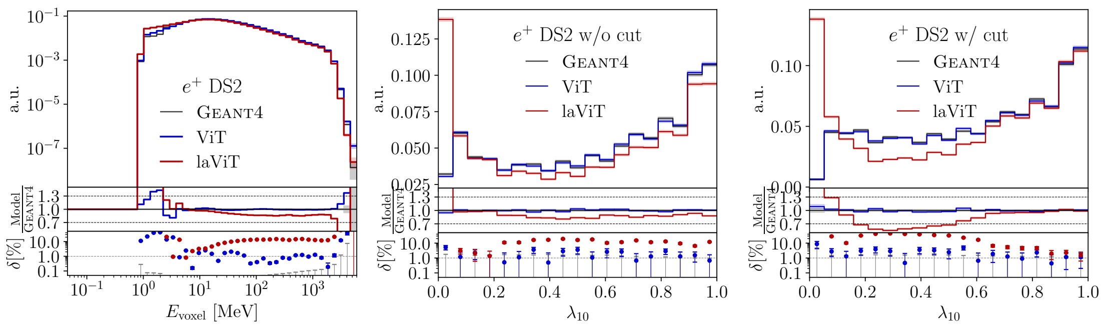


Figure 4.17: Effect of an additional threshold $E > 1$ MeV on DS2; we show the shower energy and the sparsities without and with threshold cut.

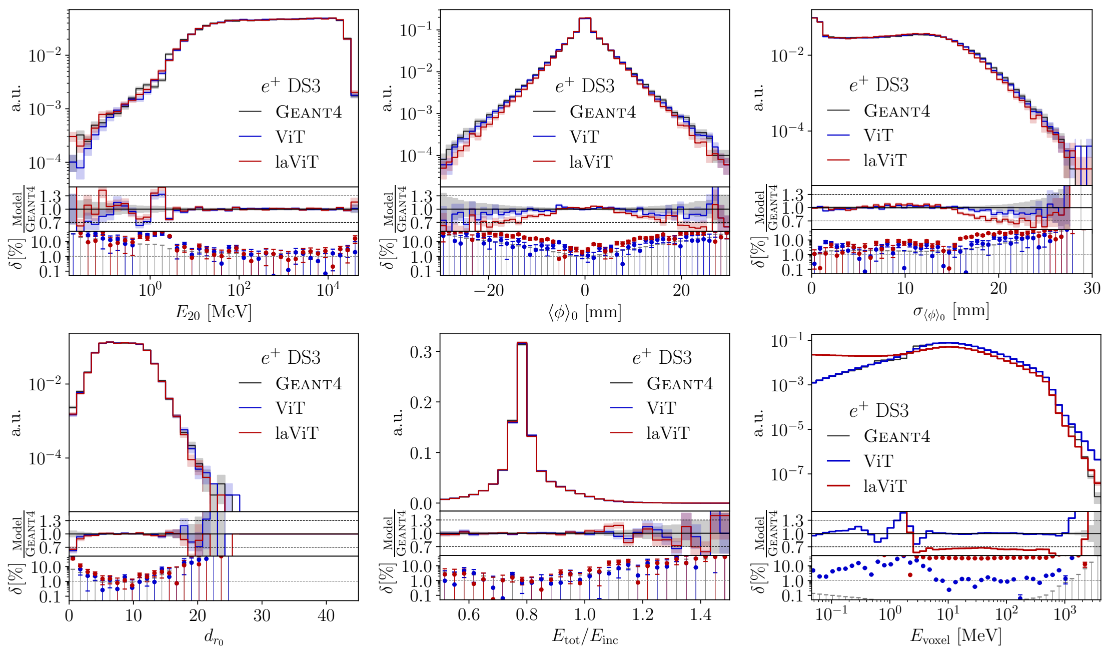


Figure 4.18: Selection of high-level features for DS3. The first row shows features for individual layers, the second row the combination of layers. All features correspond to the DS2 results shown in Fig. 4.16.

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

$$E_{\text{voxel}} > 1 \text{ MeV} . \tag{4.24}$$

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 network generates a sizable number of showers with energy depositions everywhere, leading to a peak at zero sparsity. This failure mode is already present in the autoencoder reconstruction. 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 Section 4.3.2 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 architectures that have been successful on 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 [126], where the architectures have to be modified significantly to cope with higher resolution.

In Fig. 4.18 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

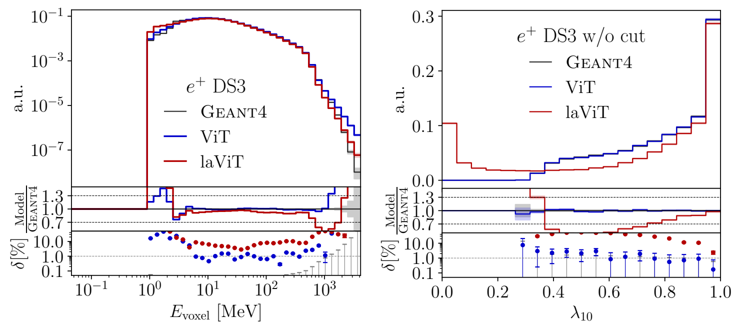


Figure 4.19: Effect of an additional threshold $E > 1$ MeV on DS3; we show the shower energy and the sparsities without and with threshold cut. All features correspond to the DS2 results shown in Fig. 4.17.

second row, we also find the same results as for DS2, including the challenge in describing voxels with $E_{\text{voxel}} \lesssim 3$ GeV.

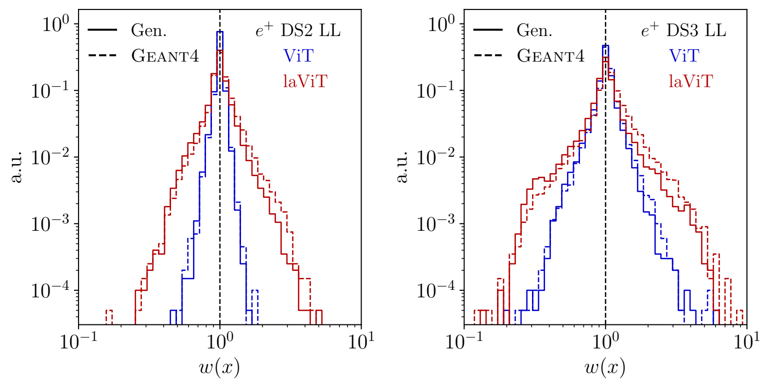
Understanding and targeting this challenge, we again show the voxel energy distribution and the sparsity after the threshold cut $E_{\text{voxel}} > 1$ MeV in Fig. 4.19. For DS3 it turns out that after applying this cut the description of DS3 through the laViT network is excellent. The reason for this 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} down 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.

Performance

It is not trivial to test the overall performance of generative networks for calorimeter showers. In the previous sections we evaluated the networks using simple one-dimensional histograms, as in Figs. 4.16 and 4.18, or classifier AUC scores. A systematic approach to assess the quality of our generative networks, and a way to identify failure modes, is to examine the distribution of classifier predictions over the phase space or feature space x [92]. A properly trained and calibrated classifier $C(x)$ learns the likelihood-ratio between the data and the generated distributions which, according to the Neyman-Pearson lemma, is the most powerful test statistic to discriminate between the two samples. This allows us to extract a correction weight over phase space

$$w(x) = \frac{C(x)}{1 - C(x)} \approx \frac{p_{\text{data}}}{p_{\text{model}}}(x) \; , \tag{4.25}$$

and to use the corresponding weight distributions as an evaluation metric. The weights have to be evaluated on the training data and on the generated data, because failure modes appear as tails in one of the two distributions [92]. For example, if we only look at the weights of generated samples, we may not identify cases where the generator suffers from mode collapse. To further analyze failure cases, we can study showers with small or large weights as a function of phase space, using the interpretable nature of phase spaces in particle physics.



	AUC (LL/HL)
	DS2
ViT	0.535(3)/0.517(3)
laViT	0.572(3)/0.536(2)
	DS3
ViT	0.63(1)/0.525(3)
laViT	0.62(1)/0.596(4)

Figure 4.20: Learned low-level (LL) 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 both high-level (HL) and low-level classifiers in each case.

In Fig. 4.20 we show the classifier weights from the low-level classifier for DS2 and for DS3. We explicitly distinguish the weights for generated samples (solid lines) and GEANT4 samples (dotted lines) obtained from the trained classifier. 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 shower depth introduced in Section 4.3.2, and the high-level AUC is similar across the two datasets. Details of the neural network classifier are listed in Tab. A.6.

To conclude, 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 provides excellent benchmarks in speed and in precision.

CHAPTER **5**

Generative Unfolding

As described in detail in Section 2.2, once the hard scattering is simulated and we obtain a so-called parton-level event, these events are propagated through each stage of the simulation chain. At the end of this chain, namely after detector simulation, we are left with detector-level events. In simulations, we have access to every intermediate step, such as the particle-level event, which corresponds to the state after hadronization but before detector effects. This means we have a deterministic pairing between, for example, a given parton-level or particle-level event and its corresponding detector-level event. Assuming that the detector simulation from a particle-level to a detector-level event is the same in data and in MC, we can exploit this pairing to correct for detector effects. While the output of the MC simulation chain defines a deterministic pairing, the physical process it models including the detector response is inherently probabilistic. In principle, corrections for hadronization and parton shower effects are also possible but are not considered in the context of this thesis.
There are numerous reasons to unfold. For example, propagating observations backwards through the simulation chain, enables testing different theories without rerunning costly detector simulations. Further, comparing observations of different experiments could enhance global fits but they are severely bottlenecked by differences in detectors, a problem unfolding could easily solve. Classical unfolding methods exist and are frequently applied in LHC analyses. However, they are typically limited to low-dimensional, binned distributions. In this chapter, we aim to reverse the simulation chain, i.e. to unfold observed detector-level events to particle-level events, using generative machine learning. Motivated by the discussion in Section 2.3, we start by looking into boosted semileptonic $t\bar{t}$-decays in Section 5.1. In particular, we unfold the hadronic top-quark decay with generative unfolding precisely and without bias. In Section 5.2, we present new developments in generative unfolding with distribution mapping techniques that were introduced in Section 3.5.

5.1 How to Unfold Top Decays

The content of this section was finalized in collaboration with Luigi Favaro, Roman Kogler, Alexander Paasch, Tilman Plehn and Dennis Schwarz. It is adapted from Ref. [5] but utilizes different simulations.

In this study we target an especially challenging unfolding task, a mass measurement and the unfolding of strongly peaked kinematic distributions, applied to hadronic decays of boosted top-quarks.
In Section 5.1.1 we describe the goal of the analysis, show the results from the classic CMS analysis [137], introduce the dataset, and sketch the basic features and the implementation of generative unfolding. We start by investigating low-dimensional unfolding in Section 5.1.3. We find that a major problem is the bias induced by the training data. It can be ameliorated as described in Section 5.1.4. Next, we show how the top mass can be measured from the unfolded distributions in Section 5.1.5, and in Section 5.1.6 we proceed to then unfold the entire top decay phase space for re-analysis. This study provides a blueprint for a CMS analysis using generative unfolding.

5.1.1 Goal and method

If we want to unfold top-decay events the main challenge is the model dependence and resulting bias when the top masses assumed for the simulated training data and the actual top mass differ. This could be taken care of iterative improvements of the unfolding network, but it will turn out that this approach is extremely challenging numerically. Instead, we follow a slightly different strategy:

1. we ensure that the bias from the top mass assumed in the simulated training data is small;
2. we infer the correct top mass from the data, using a reduced unfolded phase space;
3. we produce training data with the inferred top mass and unfold the full phase space.

Top mass measurement

The extraction of the top mass from the invariant jet mass of highly boosted hadronic top decays can shed light on possible ambiguities in top mass measurements using simulated parton showers. The ultimate goal is to compare the measured jet mass distribution to predictions from analytic calculations. For that, it is convenient to unfold detector effects.

Unfolding uses simulated data, biasing the unfolded data towards the model used in the simulation. In particular, the choice of the top mass in the simulation leads to a significant uncertainty. These modelling biases can be reduced by including more information and granularity into the unfolding process, motivating the use of ML-unfolding methods.

In the existing CMS measurement this is done by unfolding differentially in the top-jet transverse momentum and by including various sideband regions close to the measurement phase space. Using ML-unfolding, the data can be unfolded in a larger number of phase space dimensions, providing ways to reduce the model bias. In addition,

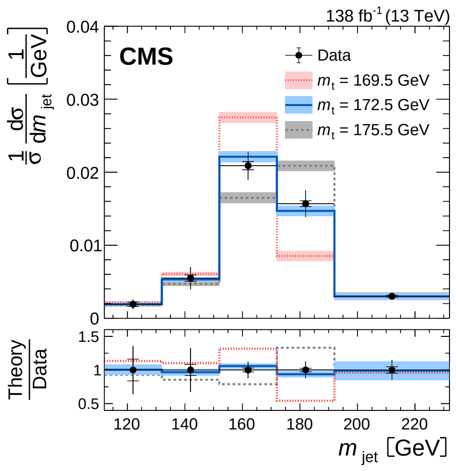


Figure 5.1: CMS benchmark result from Ref. [137]. It shows the differential top pair cross section as a function of the top-jet invariant mass, compared to theory predictions for different top masses. The vertical bars represent the total uncertainties, statistical uncertainties are shown as short horizontal bars, and theoretical uncertainties as shaded bands.

externalized measurements of the N-subjettiness and the W-mass, which were used to constrain uncertainties in the modelling of final state radiation and the jet mass scale can be included through high-dimensional ML-unfolding.

The result from our traditional benchmark analysis [137] is shown in Fig. 5.1. This analysis unfolds the reconstructed 3-subjet mass M_{jjj} and the corresponding reconstructed transverse momentum, $p_{T,jjj}$ to measure the top mass. Before that, the jet energy is calibrated by reconstructing the W-boson, which comes with non-negligible uncertainties. First, reconstructing the W-boson requires b-tagging information, which is not available in the XCone jet algorithm, so the jet clustering needs to be rerun after including b-tagging information matched by angular closeness. Second, selecting the W-decay jets breaks the permutation invariance among the jets. Third, mis-identification as part of the b-tagging introduces a non-trivial uncertainty in the unfolded data. Ideally, unfolding enough phase space dimensions to capture the W-decays should allow us to circumvent these issues.

Once we have measured the top mass in an event sample, we can further analyse the unfolded dataset. For instance, we can look for effects from higher-dimensional SMEFT operators on the decay of boosted tops, or we can search for anomalous kinematic distributions from new particles, modified interactions, or enhanced QCD effects at the subjet level. While the unfolding for the top mass measurement has to include a sufficiently large number of dimensions, as discussed above, we now need to unfold the full, 12-dimensional phase space. Three of these dimensions are finite jet masses, generated by QCD effects.

Dataset

We use simulated events for top pair production, similar to the one used for the CMS jet mass measurement [137]. It is split into three parts, corresponding to data taking in 2016, 2017, and 2018. All parts are generated at next-to-leading order QCD using

POWHEG v2 [122, 123]. The simulation for 2016 uses NNPDF3.0 [138], while the corresponding parts for 2017 and 2018 use NNPDF3.1 [116]. Hadronization, parton showers, and multiple parton interactions are simulated with Pythia8 [25]. The 2016 (2017 and 2018) sample uses the underlying event tune CUETP8M2T4 (CP5) as implemented in Pythia 8.212 (Pythia 8.230). All samples include a full simulation of the CMS detector [45] implemented in Geant4 [139].

In the simulated data, we have access to three different levels of particles. The parton level includes the hard interactions of the top quarks, that decay into a b-quark and a W-boson, that subsequently decays into two quarks or lepton and neutrino. The particle level refers to all stable particles with lifetimes longer than 10^{-8} s after parton shower and hadronization. Finally, the detector level describes particle candidates after the detector simulation, as reconstructed from tracks and energy deposits in the calorimeters.

Event selections are applied at the particle and detector level. All events that do not pass any of the selections are rejected from the further analysis. For the signal or measurement region, we only consider lepton-hadron top pairs at the parton level,

$$pp \to t\bar{t} \to (bq\bar{q}')\ (\bar{b}\ell^-\bar{\nu}) + \text{c.c.} \quad \text{with} \quad \ell = e, \mu \ , \tag{5.1}$$

and the lepton acceptance

$$p_{T,\ell} > 60 \text{ GeV} \quad \text{and} \quad |\eta_\ell| < 2.4 \ . \tag{5.2}$$

The top jet is constructed using XCone clustering and identified by the larger angular distance to the lepton and must fulfill

$$p_{T,J} > 400 \text{ GeV} \qquad \text{and} \qquad p_{T,j_{1,2,3}} > 30 \text{ GeV} \qquad |\eta_{j_{1,2,3}}| < 2.5 \ , \tag{5.3}$$

for the fat jet J and three subjets j. In the following, we will refer to these subjets as jets. The second jet has to have $p_T > 10$ GeV to reject poorly reconstructed events. Reducing the contribution from events where not the full top-quark decay is reconstructed within the fat jet, we require the invariant mass of the three jets, M_{jjj}, to exceed the invariant mass of the second jet and the lepton.

At the detector level, events have to satisfy similar criteria, described in detail in Ref. [137]. In addition, we require missing transverse momentum above 50 GeV, at least one b-tagged jet, and a specialized lepton isolation criterion to reduce background contributions. The measurement-region selection criteria leaves us with approximately 700.000 events, of which we use 75% for training. All events contain gen-level and reco-level distributions and are weighted to correct for data-MC differences, jet calibration etc. The XCone algorithm clusters the jets separately for reco-level jets and gen-level jets. The clustered jets are sorted according to p_T. Due to pile up, which is only added at reco-level, the leading jet at reco-level might not correspond to the leading jet at gen-level. Therefore, we evaluate the angular distances between reco-level and gen-level jets and match them if

$$\Delta R_{\text{gen,reco}} < 0.2 \ . \tag{5.4}$$

For around 15% of events, this matching does not respect the p_T-ordering at the two levels.

Jet-mass features

For the generative unfolding algorithm a perfect matching between reco-level and gen-level jets is not critical, as the reco-level is used only as a condition. We have checked that when permuting the ordering of the reco-level jets randomly, we observe no difference in performance. Once we switch to the 4-momentum representation (m, p_T, ϕ, η), we see clear differences between reco-level and gen-level, for instance in the jet masses shown in Fig. 5.2.

Differences in the jet masses are mostly due to pile-up, which in our simulation is added at reco-level. Events are to some degree corrected by removing tracks originating from pile-up vertices. The remaining difference in the jet mass mostly comes from neutral hadrons in the pile-up. This positive contribution to the jet masses is largest for the leading jet because of its largest p_T. Figure 5.2 implies that unfolding detector effects includes unfolding these pile-up effects.

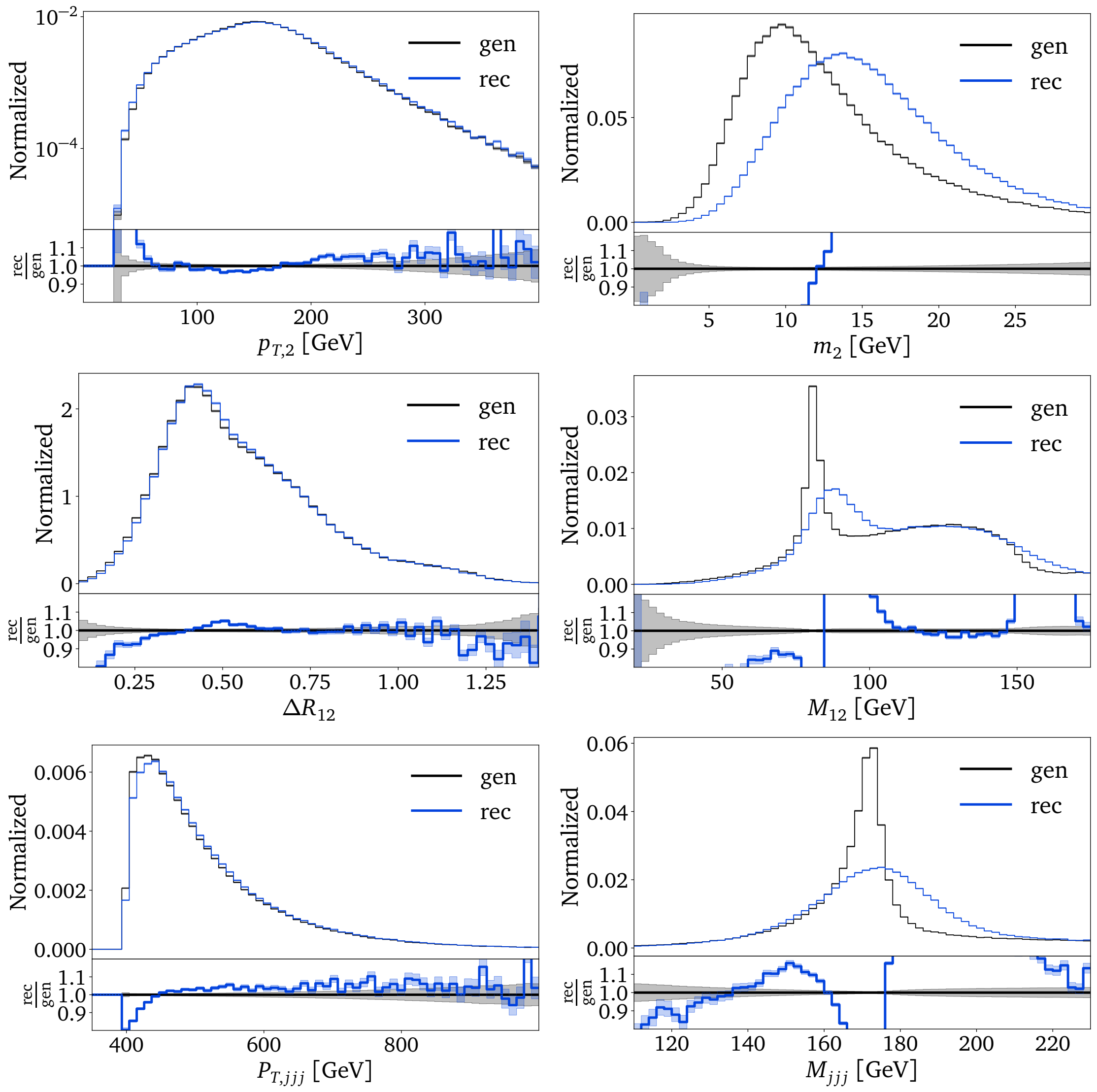


Figure 5.2: Kinematic distributions at reco-level and gen-level for the second jet (top), combining two jets (center), and combining three jets (bottom).

Going beyond single-jet observables, we need to understand and eventually unfold detector effects on correlations. In Fig. 5.2 we show a few examples. For the angular separation of the two leading jets the peak is generated by the boosted decay kinematics combined with mass effects and the detector acceptance. The 2-jet masses have a peculiar distribution, owed to the fact that out of the three jets two come from the W-decay. Because of the p_T-ordering, any of the three combinations

$$M_{ik}^2 = m_i^2 + m_k^2 + 2\ \left(m_{T,i} m_{T,k} \cosh \Delta\eta_{ik} - p_{T,i} p_{T,k} \cos \Delta\phi_{ik}\right) \tag{5.5}$$

can reconstruct m_W. This is an exact equation for the three 2-jet masses, if $\Delta\eta_{ik}$ is calculated using the jet rapidities. Of the three 2-jet masses in a top decay, two tend to be similarly close to $M_{ik} \sim m_W$ [140]. In Fig. 5.2 we also observe the upper endpoint in the top decay kinematics at gen-level [141]

$$m_{bj}^{\max} < \sqrt{m_t^2 - m_W^2} \approx 155 \text{ GeV} . \tag{5.6}$$

Following Eq.(5.5) we can improve the training of the unfolding network by including the 2-jet masses as explicit features. Each of the 2-jet masses then substitutes an angular variable. With this basis transformation we sacrifice access to the individual azimuthal angles and are left with their absolute differences.

Next, we see in Fig. 5.2 that the transverse top momentum is not affected significantly by the detector effects, and the 3-jet mass peaks at the top mass value. In our phase space parametrization we can calculate the 3-jet mass as

$$M_{jjj}^2 = M_{12}^2 + M_{23}^2 + M_{13}^2 - m_1^2 - m_2^2 - m_3^2 \tag{5.7}$$

By using all these jet masses as training features, we can greatly improve the learning and unfolding of the 3-jet mass. The no-free-lunch theorem, however, tells us that this gain will lead to a mis-modelling of other correlations. In particular, we will see that there is no guarantee that $\cos \Delta\phi \in [0, 1]$ anymore, leading to unphysical event kinematics.

5.1.2 Generative unfolding

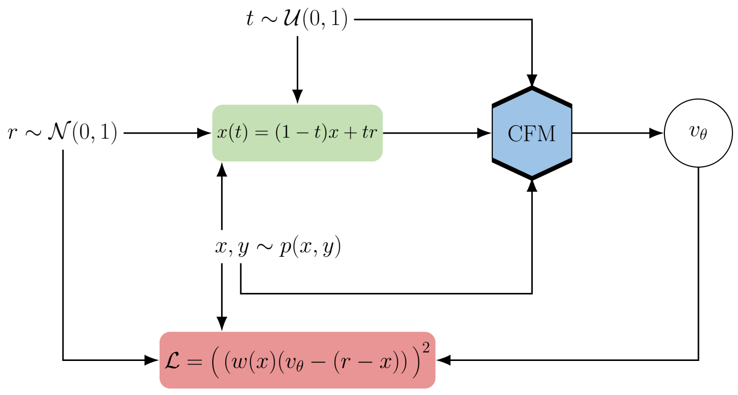
Traditional unfolding algorithms [142–144] have been used to unfold simple differential cross section measurements. Widely used methods include Iterative Bayesian Unfolding [145–148], Singular Value Decomposition [149], and TUnfold [150]. Their limitation is the need for binned data over low-dimensional phases space. This also means that we have to pre-select the observables we want to unfold and their binning.

To use ML-methods for high-dimensional and unbinned unfolding, we invert the forward simulation using Bayes' theorem

$$p(x_\text{gen}|x_\text{reco}) = p(x_\text{reco}|x_\text{gen})\, \frac{w(x_\text{gen}) p(x_\text{gen})}{w(x_\text{reco}) p(x_\text{reco})} , \tag{5.8}$$

where x_gen is a point in gen-level phase space and x_reco a point in reco-level phase space. To unfold reco-level data, we need to learn

$$p_\theta(x_\text{gen}|x_\text{reco}) \approx p(x_\text{gen}|x_\text{reco}) \tag{5.9}$$



$t \sim \mathcal{U}(0,1)$
$r \sim \mathcal{N}(0,1)$
$x(t) = (1-t)x + tr$
CFM
v_θ
$x, y \sim p(x,y)$
$\mathcal{L} = \left((w(x)(v_\theta - (r-x))\right)^2$

Figure 5.3: Schematic representation of generative unfolding with a CFM network.

as the statistical basis of an inverse simulation. Once a generative neural network encodes $p_\theta(x_\text{gen}|x_\text{reco})$, we calculate

$$p_\text{unfold}(x_\text{gen}) = \int dx_\text{reco}\; p_\theta(x_\text{gen}|x_\text{reco}) w(x_\text{reco}) p(x_\text{reco}) \; . \tag{5.10}$$

At the event level, this integral can easily be evaluated by marginalizing the corresponding joint probability. Our method can be summarized as

$$\begin{tikzcd} p_\text{sim}(x_\text{gen}) \arrow[d, leftrightarrow, "\text{paired data}"'] & & p_\text{unfold}(x_\text{gen}) \\ p_\text{sim}(x_\text{reco}) \arrow[rr, leftrightarrow, "\text{correspondence}"] & & p_\text{data}(x_\text{reco}) \arrow[u, "p_\theta(x_\text{gen}|x_\text{reco})"'] \end{tikzcd} \tag{5.11}$$

The two distributions $p_\text{sim}(x_\text{reco})$ and $p_\text{sim}(x_\text{gen})$ are encoded in one set of simulated events, before and after detector effects, or at the parton level and at reco-level.

The generative network we employ to learn $p_\theta(x_\text{gen}|x_\text{reco})$ is a CFM as dicussed in Section 3.4. As we consider weighted events the corresponding MSE loss needs to be updated to

$$\mathcal{L}_\text{CFM} = \left[w(x)(v_\theta - (r-x))\right]^2 \; . \tag{5.12}$$

The CFM setup is illustrated in Fig. 5.3. Its conditional extension is straightforward, in complete analogy to the conditional GANs [151] and conditional INNs [152] developed for unfolding. While the naive GAN setup does not learn the event-wise (inverse) migration correctly and therefore does not encode physical, calibrated conditional probabilities, the cINN with its likelihood loss does exactly that. The CFM succeeds because of its mathematical foundation, Eq.(3.39) [78].

Training bias

In Eq.(5.11) we describe the structure of generative unfolding, but we are missing a critical complication — the simulated reco-level data $p_\text{sim}(x_\text{reco})$ might not agree with the actual reco-level data $p_\text{data}(x_\text{reco})$. Let us assume a simple case where the simulation depends on a simulation parameter m_s which we can tune to describe the actual data. This can be a physics parameter we eventually infer, or a nuisance parameter which we profile over. The dependencies of the four datasets on m_s and its 'correct' value in the data, m_d, turn Eq.(5.11) into

$$\begin{tikzcd}
p_\text{sim}(x_\text{gen}|m_s) \arrow[d, "p(x_\text{reco}|x_\text{gen})"'] & p_\text{unfold}(x_\text{gen}|m_s, m_d) \\
p_\text{sim}(x_\text{reco}|m_s) \arrow[r, leftrightarrow, "\text{correspondence}"] & p_\text{data}(x_\text{reco}|m_d) \arrow[u, "p_\theta(x_\text{gen}|x_\text{reco}, m_s)"']
\end{tikzcd} \tag{5.13}$$

In the forward direction, $p(x_\text{reco}|x_\text{gen})$ does not have an explicit m_s-dependence, but both simulated datasets follow $p_\text{sim}(x_\text{gen}|m)$ and $p_\text{sim}(x_\text{reco}|m)$ induced by the generator settings. By assumption, $m_s = m_d$ ensures that the simulated and actual data agree at the reco-level,

$$p_\text{sim}(x_\text{reco}|m_s = m_d) \overset{!}{=} p_\text{data}(x_\text{reco}|m_d) \; . \tag{5.14}$$

We then use this relation to infer m_d at the reco-level.

Alternatively, we can do the same inference at the gen-level, requiring

$$p_\text{sim}(x_\text{gen}|m_s = m_d) \overset{!}{=} p_\text{unfold}(x_\text{gen}|m_s = m_d, m_d) \; . \tag{5.15}$$

The problem with this unfolded inference is the dual dependence of $p_\text{unfold}(x_\text{gen}|m_s, m_d)$ through the reco-level data and the learned conditional probability. This dual dependence is automatically resolved if $p_\text{unfold}(x_\text{gen})$ only depends on m_d through the reco-level data, so the bias from $p_\theta(x_\text{gen}|x_\text{reco}, m_s)$ can be neglected. If not, we can use iterative methods [153] to remove the bias. The iterative improvement relies on a learned classifier over x_gen which reweights p_sim to p_unfold including the m_s-dependencies and serves as a basis for re-training the unfolding network. It implicitly assumes that $p_\text{unfold}(x_\text{gen}|m_s, m_d)$ depends mostly on m_d and at a reduced level on m_s. In that case the endpoint of the Bayesian iteration is reached when the two dependencies coincide at the level of the remaining statistical uncertainty.

5.1.3 Lower-dimensional unfolding

Unfolding top decays is technically challenging, because the top mass and the W-mass are dominant features over an altogether 12-dimensional phase space. We start with a naive unfolding in, using our appropriate phase space parametrization with reduced dimensionality [154].

We know that the precision of learned phase space distribution using neural networks scales unfavorably with the phase space dimension [155, 156].[2] The full 12-dimensional phase space will not be the optimal representation to measure the top mass. Instead, we only use a lower-dimensional phase space representation for the top mass measurement, finding a balance between relevant kinematic information and dimensionality. We postpone the full kinematic unfolding to the point where we need to access to the full kinematics and benefit from the measured top mass

[2]For a possible improvement see Ref. [157].

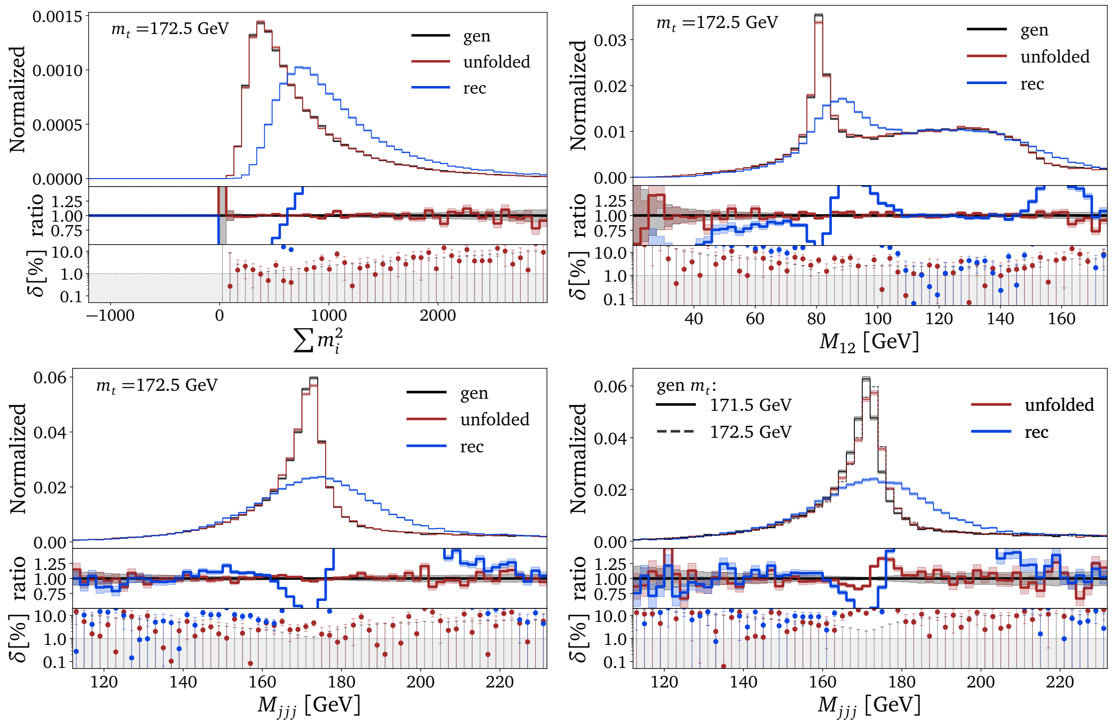


Figure 5.4: Kinematic distributions from 4-dimensional unfolding. We also show the reco-level and the gen-level truth for $m_t = 172.5$ GeV. In the bottom-right panel we compare M_{jjj} for $m_t = 172.5$ GeV to generated unfolding for $m_t = 171.5$ GeV, not seen during training.

For the traditional CMS analysis [137], two phase space dimensions were unfolded, M_{jjj} and $p_{T,jjj}$. An in-situ jet energy calibration relies on the reconstructed W-boson. Identifying the W-decay jets in the top ideally requires b-tagging information, which is not available in our case. Without b-tagging, our goal is to calibrate jet using as much reliable jet information as possible. From Fig. 5.2 we know that all 2-jet masses include a sharp W-mass peak, suggesting that we unfold those for the top mass measurement.

Our unfolding setup follows Section 5.1.1. From Eq.(5.7) we know that we can extract the 3-jet mass as a proxy for the top mass from the set of single-jet and 2-jet masses. Because the single-jet masses are largely universal and not a good handle on the jet energy calibration, our first choice is to measure the top mass from a 4-dimensional unfolding of

$$\left\{ M_{j1j2}, M_{j2j3}, M_{j1j3}, \sum_i m_i \right\} . \tag{5.16}$$

The results are shown in Fig. 5.4. First, we see that we can unfold the sum of the single jet masses extremely well, with deviations of the unfolded data from the generator truth at the per-cent level. This means that we expect to be able to extract the 3-jet mass essentially from the sum of all 2-jet masses with a known and controlled offset.

Next, we show a 2-jet mass, with the characteristic W-peak and the shoulder at $m_{bj}^{\max}$. The W-peak is washed out at the reco-level, but the generative unfolding reproduces it extremely well. The relative deviation of the unfolded to the truth 2-jet mass distributions is at most a few per-cent, with no visible shift around the W-peak. The same quality of

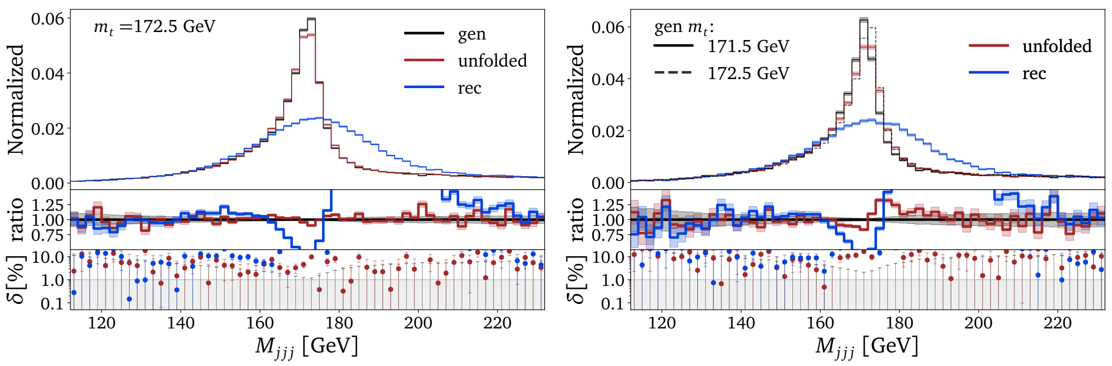


Figure 5.5: Kinematic distributions from 6-dimensional unfolding. In the right panel we compare M_{jjj} for $m_t = 172.5$ GeV to generated unfolding for $m_t = 171.5$ GeV, not seen during training.

the unfolding can be observed in the M_{jjj} distribution, perfectly reproducing the top mass at $m_t = 172.5$ GeV, the correct value in the training data and in the data which gets unfolded.

The problem with measuring the top mass from unfolded data appears when we unfold data simulated with a different top mass. In the lower-right panel of Fig. 5.4 we show the unfolded M_{jjj} distribution for reco-level data generated with $m_t = 171.5$ GeV, unfolded with generative networks trained on $m_t = 172.5$ GeV. We see that the top peak in the unfolded data is dominated by the training bias of the network, specifically a maximum at $M_{jjj} = (173 \pm 1)$ GeV. This means the top peak is entirely determined by the training bias and hardly impacted by the reco-level data which we unfold.

From the 4-dimensional unfolding we know that the network learns the W-peak in the 2-jet masses and the top peak in the 3-jet mass at a precision much below the physical particle widths. The problem is that the bias from the network training completely determines the position of these mass peaks in the unfolded data. To confirm that these findings are not an artifact of our reduced phase space dimensionality, we repeat the same analysis for the 6-dimensional phase space

$$\left\{ M_{j1j2}, M_{j2j3}, M_{j1j3}, m_{j1}, m_{j2}, m_{j3} \right\} . \tag{5.17}$$

The unfolded 3-jet mass distributions are shown in Fig. 5.5, corresponding to the 4-dimensional case in Fig. 5.4. While the unfolded peak in M_{jjj} is a slight bit worse than for the easier 4-dimensional case, the bias from the training remains, in spite of the fact that we are weakening the expressive power of the unfolding network by adding distributions that are mildly affected by the peak position.

Finally, we can look at the truth and learned migration between the reco-level and the gen-level 3-jet distribution in Fig. 5.6. In the left panel we see that the forward simulation maps the sharp peak at gen-level to a broader peak at reco-level. The problem with the central ellipse describing this physical migration by detector effects is that it does not indicate any correlation between the M_{jjj}-values at reco-level and at gen-level. The learned migration in the right panel reproduces the forward migration exactly.

For the generative unfolding this means that small differences at reco-level will always be unfolded to the same sharp region at gen-level, independent of the information contained in the reco-level data. Following Section 5.1.2 and Eq.(5.13) the unfolded distribution $p_\text{unfold}(x_\text{gen})$ is entirely determined by the training choice m_s and shows practically no dependence on the value m_d encoded in the actual data. All hyperparameters of the network training are listed in Tab. A.8.

5.1.4 Taming the training bias

The next question is how we can improve the situation where, m_s being the top mass value used for the simulation and m_d the actual top mass in the data, Eq.(5.13) turns into

$$\begin{tikzcd}
p_\text{sim}(x_\text{gen}|m_s) \arrow[d, "p(x_\text{reco}|x_\text{gen})"'] & & p_\text{unfold}(x_\text{gen}|m_s, \cancel{m_d}) \\
p_\text{sim}(x_\text{reco}|m_s) \arrow[rr, leftrightarrow, "\text{correspondence}"] & & p_\text{data}(x_\text{reco}|m_d) \arrow[u, "p_\theta(x_\text{gen}|x_\text{reco}, m_s)"']
\end{tikzcd} \tag{5.18}$$

In the unfolded distribution the training information m_s completely overwrites m_d. Moreover, even if there was enough sensitivity, a classifier comparing two shifted mass peaks learns weights far away from unity, leading to numerical challenges. This means we cannot use the usual iterative methods to remove the bias from the training data.

Following the strategy from Section 5.1.1, we first increase the sensitivity on m_d. For this, we pre-process the data such that m_d is directly accessible by adding an estimator of m_d to the representation of x_reco. Ideally, this estimator would be inspired by an optimal observable. Such a 1-dimensional sufficient statistics should exist, and we know how to construct it. For the top mass we just use the weighted median of the 3-jet masses at reco-level, M_{jjj}^batch. For a batch size around 10^4 events, this information will be strongly correlated with the top mass,

$$M_{jjj}^\text{batch} \approx m_d \equiv m_t \Big|_\text{data} \, . \tag{5.19}$$

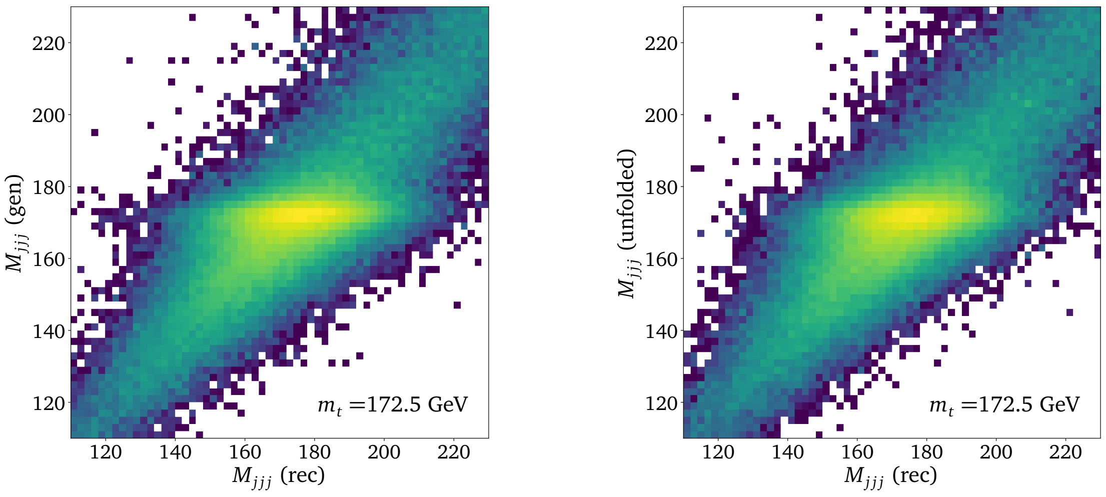


Figure 5.6: Truth and learned migration in the M_{jjj} distribution between reco-level and gen-level.

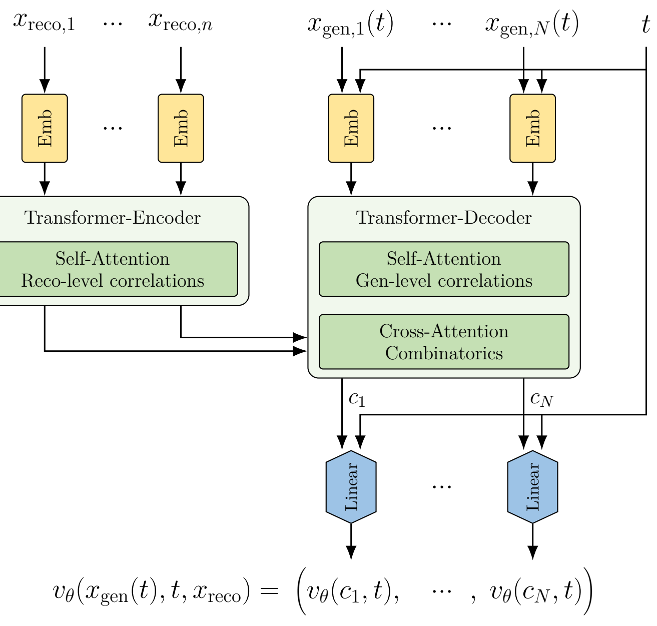

$x_{\text{reco},1}$ ⋯ $x_{\text{reco},n}$
$x_{\text{gen},1}(t)$ ⋯ $x_{\text{gen},N}(t)$
t
Emb
Transformer-Encoder
Self-Attention
Reco-level correlations
Transformer-Decoder
Self-Attention
Gen-level correlations
Cross-Attention
Combinatorics
c_1
c_N
Linear
$v_\theta(x_{\text{gen}}(t), t, x_{\text{reco}}) = \left(v_\theta(c_1, t), \quad \cdots \quad , v_\theta(c_N, t)\right)$

Figure 5.7: Schematic representation of a parallel transfusion network, adapted from [78].

This batch-wise kinematic information can be extracted at the level of the loss evaluation, and it goes beyond the usual single-event information, similar to established MMD loss modifications of GAN training [93, 151].

Second, we weaken the bias from the training data by combining training data with different top masses, but without an additional label,

$$m_t = \{169.5, 172.5, 175.5\} \text{ GeV} \qquad \text{(combined training)}. \tag{5.20}$$

It turns out sufficient to cover a range of top masses with separate, unmixed training batches. The range ensures that top masses in the actual data are within the range of the training data. We ensure a balanced training by enlarging the event samples with $m_t = 169.5$ GeV and $m_t = 175.5$ GeV to match the size of the largest sample. This is done by repeating and shuffling the input data, which effectively uses these events several times per epoch. With an appropriate regularization we avoid overfitting. The limited number of events in the simulations makes this training strategy sub-optimal. We expect larger and additional m_t simulations, unavailable at this time, to improve the results. We observed that both steps need to be included to ensure precise, unbiased results.

Obviously, this strategy of strengthening the dependence on m_d and reducing the training bias is not applicable to all problems, and it does not lead to the endpoint of the Bayesian iterative method, but for our combined inference-unfolding strategy it works, and this is all we need.

Transfusion architecture

As the network task becomes significantly more difficult we replace the simple dense architecture with a transfusion network, described in detail in Ref. [37, 78] and visualized in Fig. 5.7.

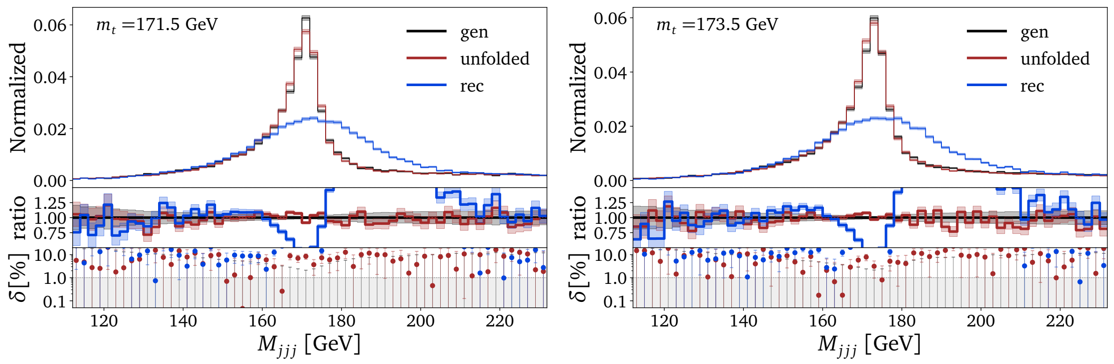


Figure 5.8: M_{jjj}-distribution from 4-dimensional unfolding of data with $m_t = 171.5$ GeV (left) and $m_t = 173.5$ GeV (right). We train the network combining samples with three top masses, Eq.(5.20).

Each component of the n-dimensional condition as well as of the time-dependent N-dimensional input $x(t)$ are individually embedded by concatenating positional information and zero padding. The embedded conditions are passed through the encoder part of a transformer, while the embedded input is passed through the decoder counterpart. In both transformer parts, we apply self-attention to learn the correlations in the condition and in the input. It is complemented by a cross-attention between encoder and decoder outputs, to learn the correlations between conditions and input. They are crucial for the unfolding task. The transformer output for every component of the input one high-dimensional embedding vector c_i, which is mapped back to a 1-dimensional component of the velocity field by a shared linear layer. This way we express the learned N-dimensional velocity field of Eq.(3.48) as

$$v_\theta(x_\text{gen}(t), t, x_\text{reco}) = (v_\theta(c_1, t), \dots, v_\theta(c_N, t)) \,. \tag{5.21}$$

The hyperparameters of the network can be found in Tab. A.9.

Using the transfusion network we unfold the 4-dimensional phase space from Eq.(5.16). The results are shown in Fig. 5.8. We unfold data generated with two different top masses, $m_t = 171.5$ GeV and $m_t = 173.5$ GeV. Neither of these two values are present in the training data. In both panels we see that the top mass as the main kinematic feature is reproduced well, without a significant effect in the relative deviation. The fitted peak values of the distribution are $m_\text{peak} = (171 \pm 1)$ GeV when unfolding data with 171.5 GeV, and $m_\text{peak} = (173 \pm 1)$ GeV when unfolding data 173.5 GeV. While the bias might not have vanished entirely, it is well contained within the numerical uncertainties. We will extract the unfolded top mass value properly in Section 5.1.5.

Dual network

Given the more complicated training task, we observe a drop in performance when we increase the dimensionality to unfold the 6-dimensional phase space

$$x = \left(\{m_i\}, \{M_{ik}\} \right) \,, \tag{5.22}$$

defined in Eq.(5.17) using the transfusion network. Inspired by Refs. [1,41], we factorize the phase space density into two parts, each encoded in a generative network: the first network learns the individual jet mass directions in phase space, which are universal and do not depend on the value of m_t; a second network generates the 2-jet masses conditioned on the individual jet masses,

$$p(x_\text{gen}|x_\text{reco}) = \underbrace{p\left(\{m_{i,\text{gen}}\}\middle|\, x_\text{reco}, M_{jjj}^\text{batch}\right)}_{\text{network 1}} \underbrace{p\left(\{M_{ik,\text{gen}}\}\middle|\, \{m_{i,\text{gen}}\}, x_\text{reco}, M_{jjj}^\text{batch}\right)}_{\text{network 2}} . \quad (5.23)$$

Both CFM-transfusion networks also receive M_{jjj}^batch calculated for a full batch using Eq.(5.7). For the event generation we first generate the unfolded jet masses $\{m_i\}$, pass them as a condition to the second network, and then generate the unfolded 2-jet masses $\{M_{ik}\}$.

Looking at the 6-dimensional correlation giving M_{jjj} in Fig. 5.9, we observe a hardly visible drop in performance, but still no bias from the training data. As before we observe peak values at $m_\text{peak} = (171 \pm 1)$ GeV when unfolding data with 171.5 GeV and at $m_\text{peak} = (173 \pm 1)$ GeV when unfolding data with 173.5 GeV.

5.1.5 Mock top-quark mass measurement

We estimate the benefit from generative unfolding by repeating the top mass measurement from Ref. [137] on the same simulated data, but with a large number of bins in the M_{jjj} histogram. The top mass is extracted from the binned unfolded distributions using a fit based on $\chi^2 = d^T V^{-1} d$, where d is the vector of bin-wise differences between the normalized unfolded distribution and the normalized prediction from the simulated data. The covariance matrix V contains the uncertainties and corresponding bin-to-bin correlations. A parabola fit provides the central value of m_t and the standard deviation.

Statistical and model uncertainties

First, this fit requires the covariance matrix describing statistical uncertainties [158]. We sample from the latent space, conditional on the reco-level events, N times, which means we generate N unfolded distributions from the posterior $p_\theta(x_\text{gen}|x_\text{reco})$. We then use a Poisson bootstrap, where we assign a weight from a Poisson distribution with unit mean. The size of one replica is 41,000 events, corresponding to the smallest training sample

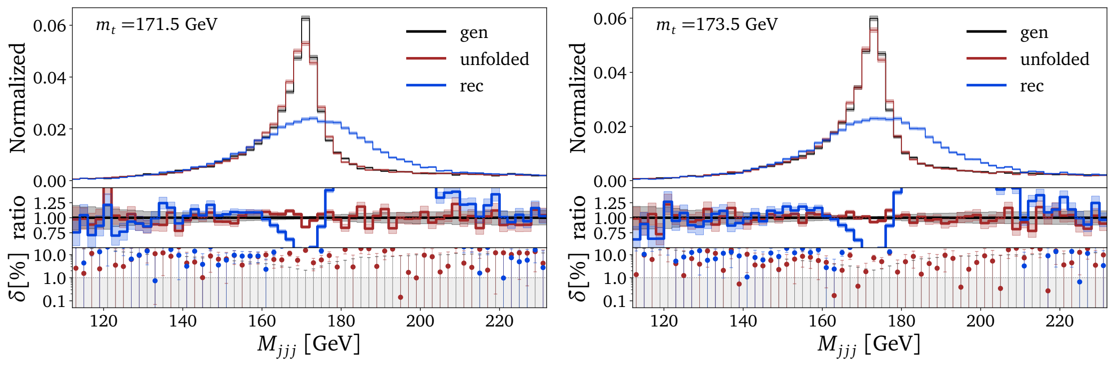


Figure 5.9: Unfolded M_{jjj} with two networks from 6-dimensional unfolding of data with m_t = 171.5 GeV (left) and m_t = 175.5 GeV (right). We train the network combining samples with three top masses, Eq.(5.20).

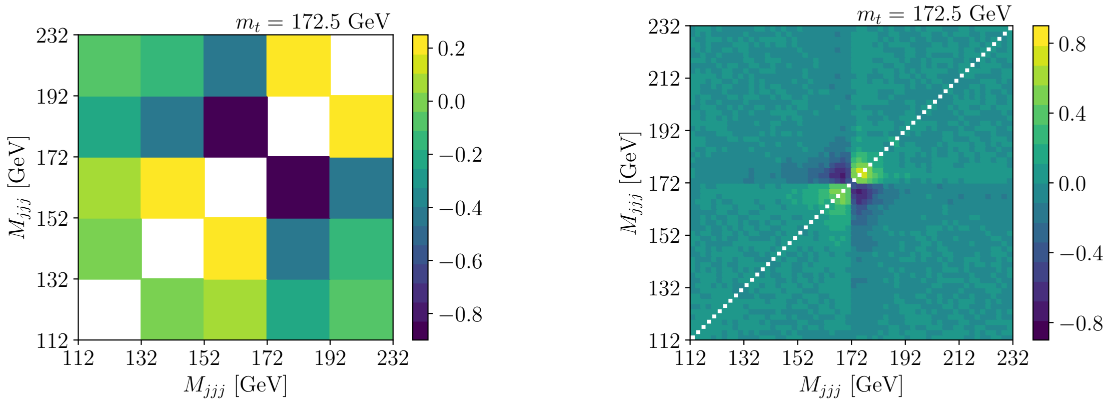


Figure 5.10: Correlation matrices for $m_t = 172.5$ GeV from $N_\text{rep} = 1000$ replicas in 60 bins (right) and 5 bins (left) for the 4-dimension unfolding.

with $m_t = 169.5$ GeV and the approximate number of real data events. The number of events follows a Poisson distribution, with the mean given by the nominal sample size.

For the measurement, we create $N_\text{rep} = 1000$ replicas by selecting the nominal number of reco-level events from the test dataset with $m_t = 172.5$ GeV and the full datasets for the simulations at different top masses. We unfold each replica, calculate M_{jjj}, and use the histogram entries $u_i^{(n)}$ to compute the correlation matrix of statistical fluctuations as

$$\begin{aligned} \text{cov}_{ij} &= \frac{1}{N_\text{rep}} \sum_{n=1}^{N_\text{rep}} (u_i^{(n)} - \bar{u}_i)(u_j^{(n)} - \bar{u}_j) \qquad \text{with} \qquad \bar{u}_i = \frac{1}{N_\text{rep}} \sum_{n=1}^{N_\text{rep}} u_i^{(n)} \\ \text{corr}_{ij} &= \frac{\text{cov}_{ij}}{\sqrt{\text{cov}_{ii}}\sqrt{\text{cov}_{jj}}} \; . \end{aligned} \tag{5.24}$$

This procedure also takes into account the uncertainties due to the statistical fluctuations of M_{jjj}^batch. The training of the network itself introduces correlations which are at least one order of magnitude smaller and therefore ignored in the measurement.

The 60×60 correlation matrix for the 4-dimensional unfolding using the largest sample generated with $m_t = 172.5$ GeV is shown in Fig. 5.10. We see two distinct source of bin-to-bin correlations. In general, an event migrating from bin i to bin j gives rise to negative correlations in the number of events between the two bins. Additionally, unbiasing the unfolding ensures that a shift in the batch-wise condition also shifts the unfolded peak. This effect, accounted for in the bootstrapping method, introduces an additional contribution to the bin-to-bin correlations. It causes positive correlations between bins on the same side of the peak and anti-correlations otherwise. In our case, both effects are most apparent in the peak region and its neighbouring bins as shown in Fig. 5.10. Second, we follow Ref. [137] to estimate the uncertainty from the choice of m_t in the simulation. We evaluate the difference between the unfolded distribution and the corresponding simulated gen-level distribution for each bin and construct a covariance matrix

$$\text{cov}_{ij} = \sigma_i \rho_{ij} \sigma_j \; , \tag{5.25}$$

where σ_i is the uncertainty in bin i, and ρ_{ij} the correlation between bins i and j. These bin-to-bin correlations are not known, but since we do not observe any systematic pattern, we neglect the bin-to-bin correlations and use a diagonal covariance matrix. It was verified that other choices do not alter the results. To estimate the impact of this model uncertainty, we perform the m_t extraction twice. First, we only include the statistical covariance matrix corresponding to 41,000 available events at the reco-level. Second, we repeat the same measurement also including the model uncertainty.

Improvement

To compare our new unfolding technique to the existing TUnfold results [137], we repeat the extraction using the simulated data sets with 172.5 GeV and using the statistical covariance matrix from the measured data, published in HEPData [159]. We scale the statistical uncertainty by a factor of $\sqrt{52/41}$ to account for the fact that the CMS measurement uses about 52,000 events. The χ^2-curves and the corresponding results are displayed in Fig. 5.11, where we show the 4-dimensional and 6-dimensional unfoldings with 5 bins and the TUnfold result. While the statistical uncertainties are slightly larger for the unbinned unfolding, we see that the uncertainty in the choice of m_t is reduced from being a leading model uncertainty in the CMS measurement to a negligible level.

To confirm that the choice in m_t does not leave a residual bias, we repeat the top-quark mass extraction for different top masses in the reco-level data. The results are shown in the left panel of Fig. 5.12. While the bin width in the unfolding with TUnfold is limited by the jet mass resolution, we test various binning schemes for the unbinned unfolding and observe closure for up to 20 bins. For finer binnings the resolution is too fine for the comparably coarse grid of available gen-level distributions with $m_t = \{169.5, 171.5, 172.5, 173.5, 175.5\}$ GeV, leading to an unstable closure test altogether.

Circumventing this limitation, we interpolate the gen-level distributions for m_t-values close to 172.5 GeV, where three samples with a separation of 1 GeV are available and a linear dependence of the bin content on m_t is a valid approximation. Now, we can compare the generative unfolding with 5 to 60 bins in terms of the statistical uncertainty. The results are displayed in the right panel of Fig. 5.12, indicating an increase in the precision of the top mass by up to 40% due to the improved resolution.

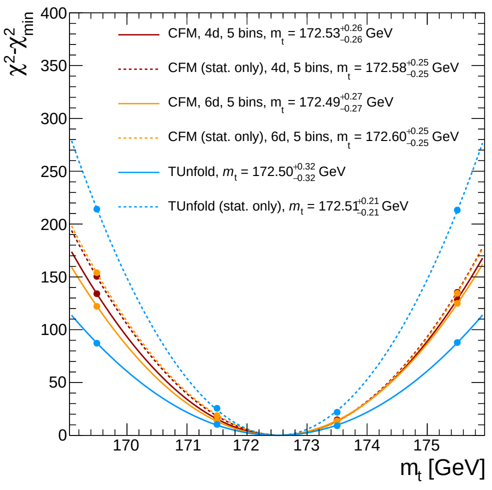


Figure 5.11: Extraction of m_t with a χ^2 test only accounting for statistical uncertainties and with the additional model uncertainty from the choice of m_t in the simulation.

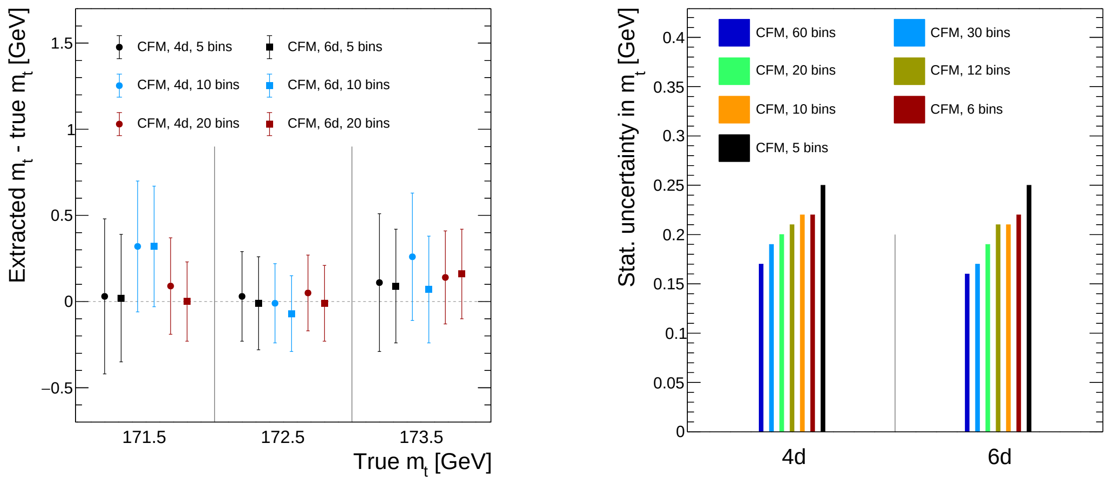


Figure 5.12: Left: deviation of the extracted top mass from the reco-level truth, employing 4-dimensional and 6-dimensional unfolding for the assumed values of 171.5, 172.5, and 173.5 GeV. Right: statistical uncertainties of the extractions of the top mass from the 4-dimensional and 6-dimensional unfolding with different binnings, assuming $m_t = 172.5$ GeV.

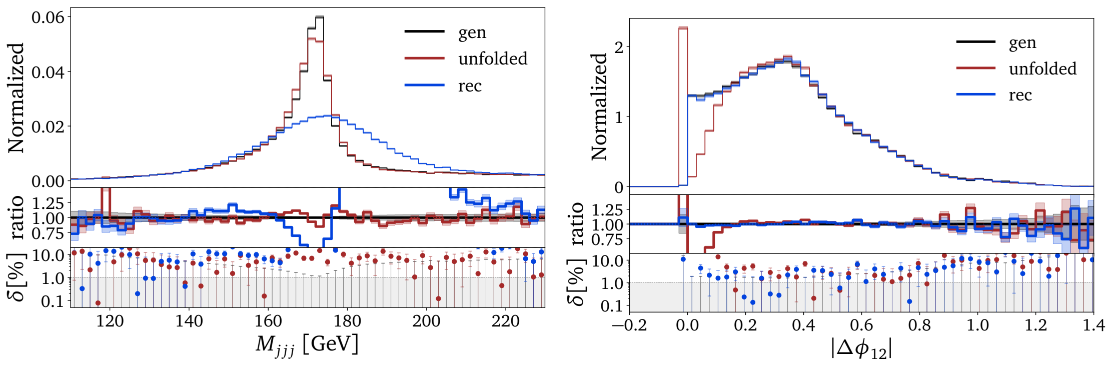


Figure 5.13: Kinematic distributions from full, 12-dimensional unfolding. We show the 3-jet mass as well as the azimuthal angle between the two leading jets.

5.1.6 Full phase space unfolding

As a last step of our unfolding program, we need to unfold the full 12-dimensional phase space given the measured top mass. This has the advantage that the leading source of training bias is removed. Following the same precision arguments as before, we keep the mass basis of Eq.(5.17) for the first 6 of the 12 phase space dimensions. This ensures that the 2-jet and 3-jet masses are reproduced well, albeit not at the level of the dedicated first unfolding step.

The remaining phase space dimensions are

$$x = \left(\{m_i\}, \{M_{ik}\}, \{p_{T,i}\}, \{\eta_i\} \right) \qquad i,k = 1,2,3 \, , \tag{5.26}$$

all other kinematic observables can be computed from these basis directions. For the 12-dimensional unfolding we use a single transfusion network, after checking that the dual network does not lead to an improvement. The hyperparameters are given in Tab. A.10.

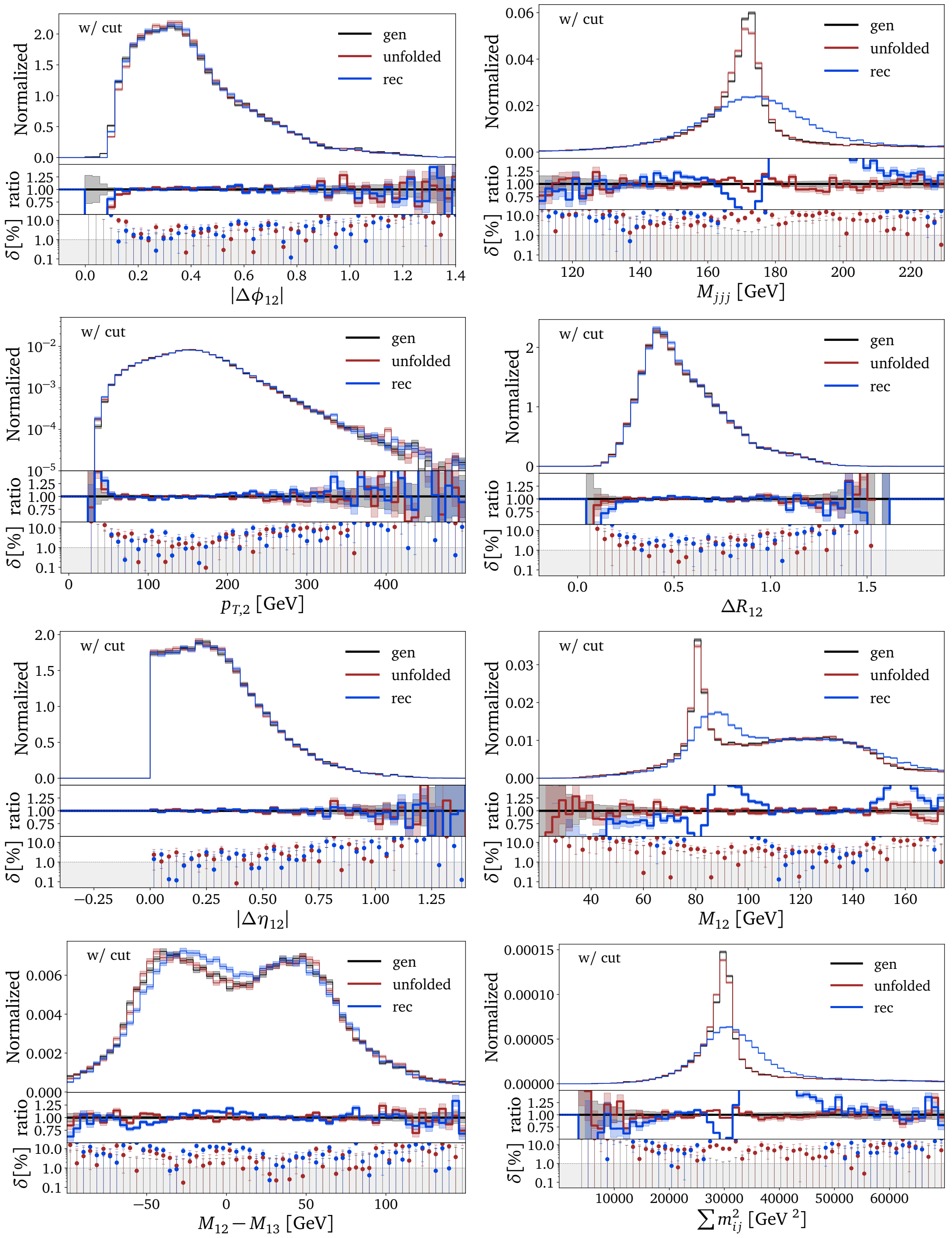


Figure 5.14: Kinematic distributions from full, 12-dimensional unfolding. We show the target 3-jet distribution, the azimuthal angle between the jets after cut, and a set of single-jet observables, 2-jet correlations, and 3-jet correlations (top to bottom).

Two kinematic distributions are shown in Fig 5.13. In the left panel we see that the top mass peak is learned almost as well as for the 4-dimensional and 6-dimensional cases. Indeed, this is the case for all jet masses and 2-jets masses, which are combined to the 3-jet mass with the top peak.

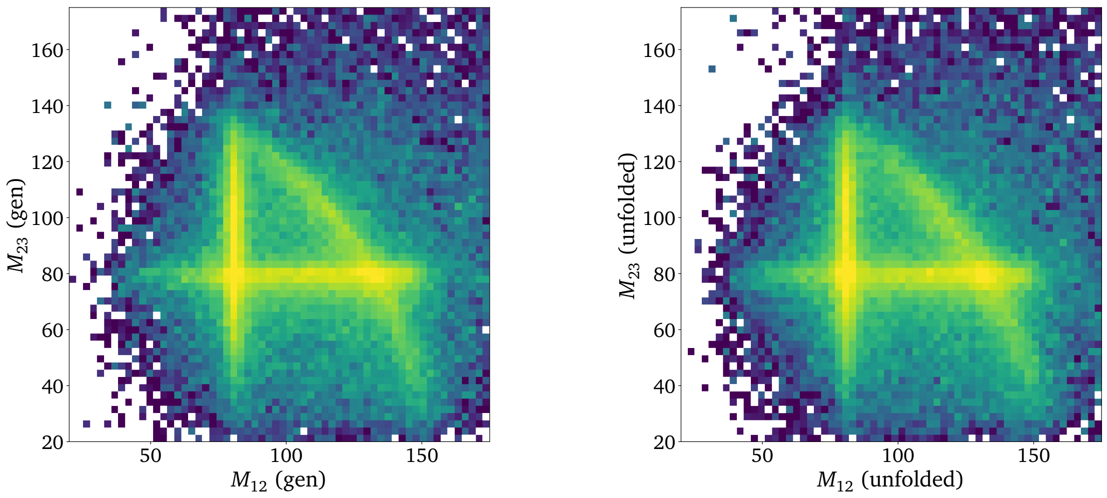


Figure 5.15: Correlation of two 2-jet masses at gen-level truth (left) and after unfolding (right).

A serious issue arises from the azimuthal angle between the two leading jets, $\Delta\phi$. According to Eq.(5.5) this angle is learned as a correlations of 7 phase space directions. Moreover, we do not have access to the azimuthal angles, only to the cosine of differences between angles. Here the problem arises that the network does not ensure that this cosine comes out in the physical range -1 ... 1. We enforce the physical range by clipping the cosine for small angles to one, which causes a mis-modelling of the small-$\Delta\phi$ regime, shown in the right panel of Fig. 5.14.

A simple way to improve this mis-modelling is to require $\cos\Delta\phi_{12} < 1$. However, from Fig. 5.13 we know that this does not solve the problem. Instead, we accept the fact that for unfolding the masses well we might have to pay a prize in the coverage of the angular correlations, and we apply an additional acceptance cut

$$\Delta\phi_{ij} > 0.1 \tag{5.27}$$

both, at the reco- and gen-levels of the unfolded events. This reduces the unfolded dataset by 30%. An extended set of unfolded kinematic distribution after this cut are shown in Fig 5.14. We know that our unfolding methods covers correlations between the original phase space directions well, because many of the kinematic observables shown in Fig 5.14 are, in reality, complex correlations of our phase space basis. However, to end with a nice figure and to drive home the message that high-dimensional unfolding using conditional generative networks does learn the corresponding correlations well, we show one of our favorite correlations in Fig 5.15. Indeed, there is literally no difference in the correlation between two of the three 2-jet masses. This correlation also confirms that the condition $M_{ik} \approx m_W$ leads to three distinct lines in phase space, where close to the crossing points it is impossible to reconstruct which of the two jets come from the W-decay.

For our study, we unfold detector effects from boosted top decay data using state-of-the-art conditional generative networks. Unfolding decay kinematics is especially challenging because we expect a large model dependence and even systematic bias from the choice of the top mass in the simulated training data. Our study shows that generative unfolding can solve this problem and provides a significant milestone towards incorporating generative unfolding in an existing LHC analysis. This study serve as a blueprint for an actual CMS analysis, both, for a top mass measurement and for a wider use of the unfolded data.

5.2 Generative Unfolding with distribution mapping

The content of this section was finalized in collaboration with Anja Butter, Sascha Diefenbacher, Nathan Huetsch, Vinicius Mikuni, Benjamin Nachman and Tilman Plehn. It is adapted from Ref. [4].

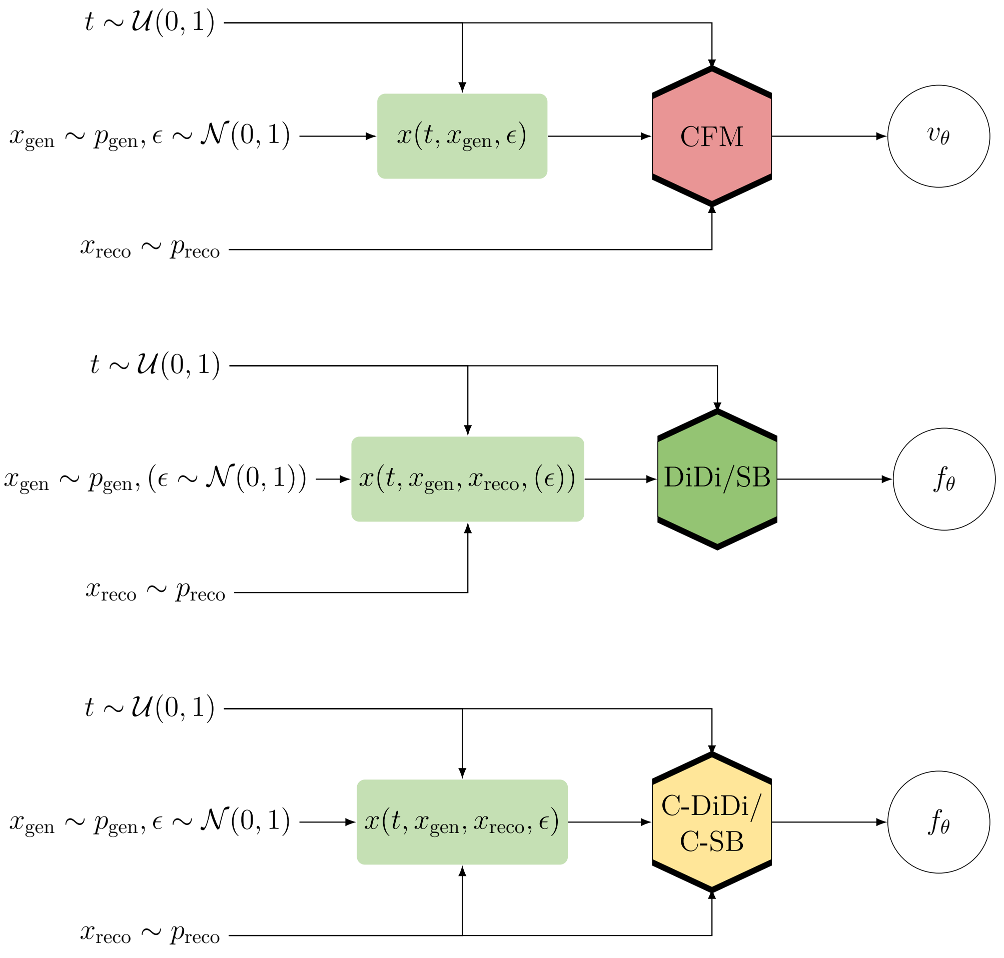
In this study, we focus on a class of generative model-based approaches that use distribution mapping, whereby the experimental events are morphed to match the corresponding gen-level events. By starting from the experimental events directly, the generative model only needs to move the events a little (assuming a precise detector), whereas other generative approaches need to map generic Gaussian random variables into the data distribution. Previously, two distribution mapping approaches were proposed, both based on conditional diffusion models [160]: one using Schrödinger Bridges (SBUnfold [77]) and one using Direct Diffusion (DiDi [78]). Previous work showed that these techniques showed excellent performance on the marginal distributions of the target cross sections, but they were not able to preserve the conditional distributions of the detector response. This could lead to a strong dependence on the gen-level simulation and is thus undesirable. The goal of this study is to remedy this issue through conditioning [82] as described in Section 3.5.3. Along the way, we introduce a new benchmark dataset, inspired by the recent ATLAS measurement [161], that can be used for distribution mapping as well as any unfolding method.

New methods following the discussing in Section 3.5 are presented in the context of unfolding in Section 5.2.1 and tested on a dataset of single jet substructure in Section 5.2.2. We then create a a new dataset describing a 22-dimensional phase space in $Z + 2$-jets at the Large Hadron Collider in Section 5.2.3. This latter dataset combines jet substructure and kinematic information. For all applications, we provide a detailed discussion of the conditional Schrödinger Bridge and Direct Diffusion performance and a comparison with the state of the art in generative unfolding, a diffusion model using transformer layers [78, 152].

5.2.1 Methodology

For unfolding, we want to transform a measured reco-level distribution p_{reco} to the corresponding gen-level distribution p_{gen}. Recent implementations [78], such as presented in Section 5.1, of generative unfolding use a CFM network to generate samples from the posterior distribution $p(x_{\text{gen}}|x_{\text{reco}})$. It learns the velocity $v(x, t, x_{\text{reco}})$, linked to the posterior distribution via Eq.(3.69) and flowing between a point of a Gaussian latent space and a point in the gen-level phase space conditioned on a given rec-level event. A schematic illustration of the training procedure is shown in the top part of Fig. 5.16.

Alternatively, we can also map reco-level events directly to their gen-level counterpart either using the SB or DiDi. The key difference to standard generative unfolding is that we use the reco-level information to define the trajectories instead of treating it as an additional input to the network. This is visualized in the center of Fig. 5.16. We learn the drift term of the probability $p(x, t)$ as described in Eq.(3.92). Optionally, we



$t \sim \mathcal{U}(0,1)$
$x_\text{gen} \sim p_\text{gen}, \epsilon \sim \mathcal{N}(0,1)$
$x(t, x_\text{gen}, \epsilon)$
CFM
v_θ
$x_\text{reco} \sim p_\text{reco}$
$t \sim \mathcal{U}(0,1)$
$x_\text{gen} \sim p_\text{gen}, (\epsilon \sim \mathcal{N}(0,1))$
$x(t, x_\text{gen}, x_\text{reco}, (\epsilon))$
DiDi/SB
f_θ
$x_\text{reco} \sim p_\text{reco}$
$t \sim \mathcal{U}(0,1)$
$x_\text{gen} \sim p_\text{gen}, \epsilon \sim \mathcal{N}(0,1)$
$x(t, x_\text{gen}, x_\text{reco}, \epsilon)$
C-DiDi/ C-SB
f_θ
$x_\text{reco} \sim p_\text{reco}$

Figure 5.16: Schematic illustration of the training procedure of a CFM-based (top), a distribution mapping-based (middle) and a conditional distribution mapping-based (bottom) generative unfolding pipeline.

can add Gaussian noise to the trajectories to make the networks stochastic rather than deterministic. The impact of the noise is governed by the choice of the diffusion term $g(t)$.

Finally, we can combine the conditional generative approach with the DM by giving the reco-level information to the network directly. The training objective is to learn a drift term linked to the conditional probability $p(x(t), t|x_\text{reco})$ as in Eq.(3.100). In this scenario adding noise is not optional. The exact training procedure is illustrated in the lower part of Fig. 5.16.

5.2.2 Unfolding Jet Substructure Observables

As a first physics example, we consider the updated version [162] of the OmniFold dataset [163] which has become a standard benchmark for unfolding methods [77,78,164]. It consists of events describing

$$pp \to Z + \text{jets} \tag{5.28}$$

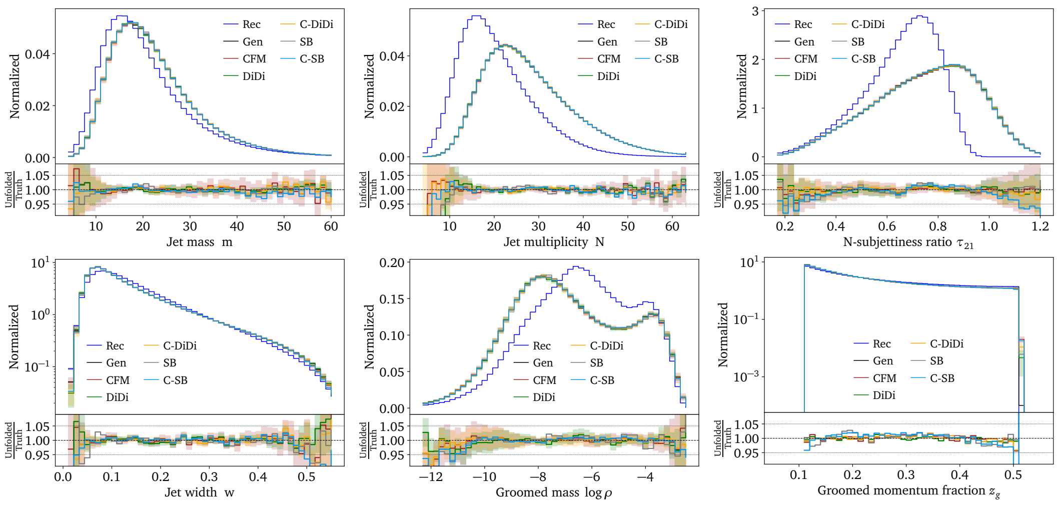


Figure 5.17: Unfolded distributions of the 6d jet-substructure dataset using CFM, DiDi, C-DiDi, SB and C-SB. All unfolded distributions reproduce the truth at percent level. The remaining differences are well covered by the BNN uncertainties.

production at $\sqrt{s} = 14$ TeV. The events are generated and decayed with Pythia 8.244 [25] with Tune 26, the detector response is simulated with DELPHES 3.5.0 [28] with the CMS card, that uses particle flow reconstruction. At both pre-detector (gen level) and post-detector (reco level) jets are clustered with the anti-k_T algorithm [50] with $R = 0.4$, as implemented in FastJet 3.3.2 [53].

We unfold six jet substructure observables of the leading jet: mass m, width $\tau_1^{(\beta=1)}$, multiplicity N, soft drop mass [165, 166] $\rho = m_{\text{SD}}^2/p_T^2$, momentum fraction z_g using $z_{\text{cut}} = 0.1$ and $\beta = 0$, and the N-subjettiness ratio $\tau_{21} = \tau_2^{(\beta=1)}/\tau_1^{(\beta=1)}$ [167]. The dataset contains about 24M simulated events, 20M for training and 4M for testing.

Unfolded distributions

We unfold the 6-dimensional phase space using all five network implementations of the three methods

- conditional generative (Conditional Flow Matching, CFM);
- unconditional distribution mapping DiDi and SB; and
- conditional distribution mapping C-DiDi and C-SB.

The respective velocity fields and drift terms are encoded in standard MLPs, the hyperparameters are given in Tab. A.11. All networks are implemented in PYTORCH [84] and trained with the ADAM [60] optimizer. We follow the preprocessing from Ref. [78].

Due to the varying numerical requirements of different networks, we choose suitable numerical solvers for their evaluation. For the CFM, an ODE-based network, we unfold reco-level samples using a numerical ODE-solver [168]. The SDE-based networks C-DiDi and C-SB use the DDPM SDE-solver [39]. For the unconditional DiDi and SB we have the choice between an ODE-based formulation with noise scale $g = 0$ and an SDE-based formulation with $g > 0$. We observe no significant difference in performance between them, the shown results use the SDE formulation.

In Fig. 5.17 we show all unfolded distribution together with the true gen-level and reco-level distributions. For CFM, DiDi and SB, we reproduce the results shown in Ref. [78]. Both methods can reliably solve this unfolding task to sub-percent precision. The new C-DiDi and C-SB give a precision on par with the established CFM method. For all networks, we only observe significant deviations from the truth far into the tails or at hard edges, for instance, in the groomed momentum fraction z_g.

The uncertainties reported in the figures are produced from posterior sampling. This is possible by making all of the models Bayesian neural networks, approximated with independent Gaussians for every network parameter, doubling the number of model parameters [67–69, 72, 169]. Previous studies in the context of the LHC have shown that the posterior is a reasonable estimate of the variation introduced by the limited size of the training dataset [71, 156, 170]. Even though the weights are Gaussian distributed, the final network output is generally not a Gaussian. The concept of BNNs can also be applied to the density estimation in generative networks [1, 41, 66], including diffusion generators [1, 2] as described in Section 3.4.2.

The uncertainties shown in Fig. 5.17 are obtained by evaluating the respective networks 20 times, each time with a new set of network weights sampled from the learned distribution. The deviations from the true gen-level distribution as well as the differences between the five networks are generally covered by these uncertainties. As expected, the uncertainty increases in regions of low training statistics e.g. the tail of the jet mass distribution m.

Learned Mapping

Next, we check if the learned mapping between the reco-level and gen-level distributions agrees with the physical forward simulation. We show migration matrices for some of the observables in Fig. 5.18: the first row shows the truth encoded in the training data; the following columns show the learned event-wise mapping from the the CFM, unconditional DiDi/SB and C-DiDi/C-SB. While the 6-dimensional unfolded distributions are nearly identical for all methods, the migration plots show a significant difference between the unconditional and conditional networks.

The conditional CFM generator is, by design, trained to reproduce the conditional distribution $p(x_\text{gen}|x_\text{reco})$. In contrast, the unconditional DM learns to map $p(x_\text{reco}) \to p(x_\text{gen})$, but with an unphysically diagonal optimal transport prescription, as showcased in the third row. Finally, we see that the conditional C-DiDi and C-SB encode the conditional probabilities just like the CFM does.

Finally, we take a closer look at the learned posterior distributions $p(x_\text{gen}|x_\text{reco})$. While single-event-unfolding is an ill-defined analysis task, we can use the per-event posterior to illustrate the performance of the different unfolding generators [36, 37]. All our methods are inherently non-deterministic when inputting the same reco-level event repeatedly, which allows us to generate non-trivial learned posteriors. For the CFM, we sample the latent Gaussian distribution, for any one given latent space point the ODE trajectory is deterministic. In contrast, the DM-methods always start from the same latent space point, the reco event, but evolve it using a non-deterministic SDE.

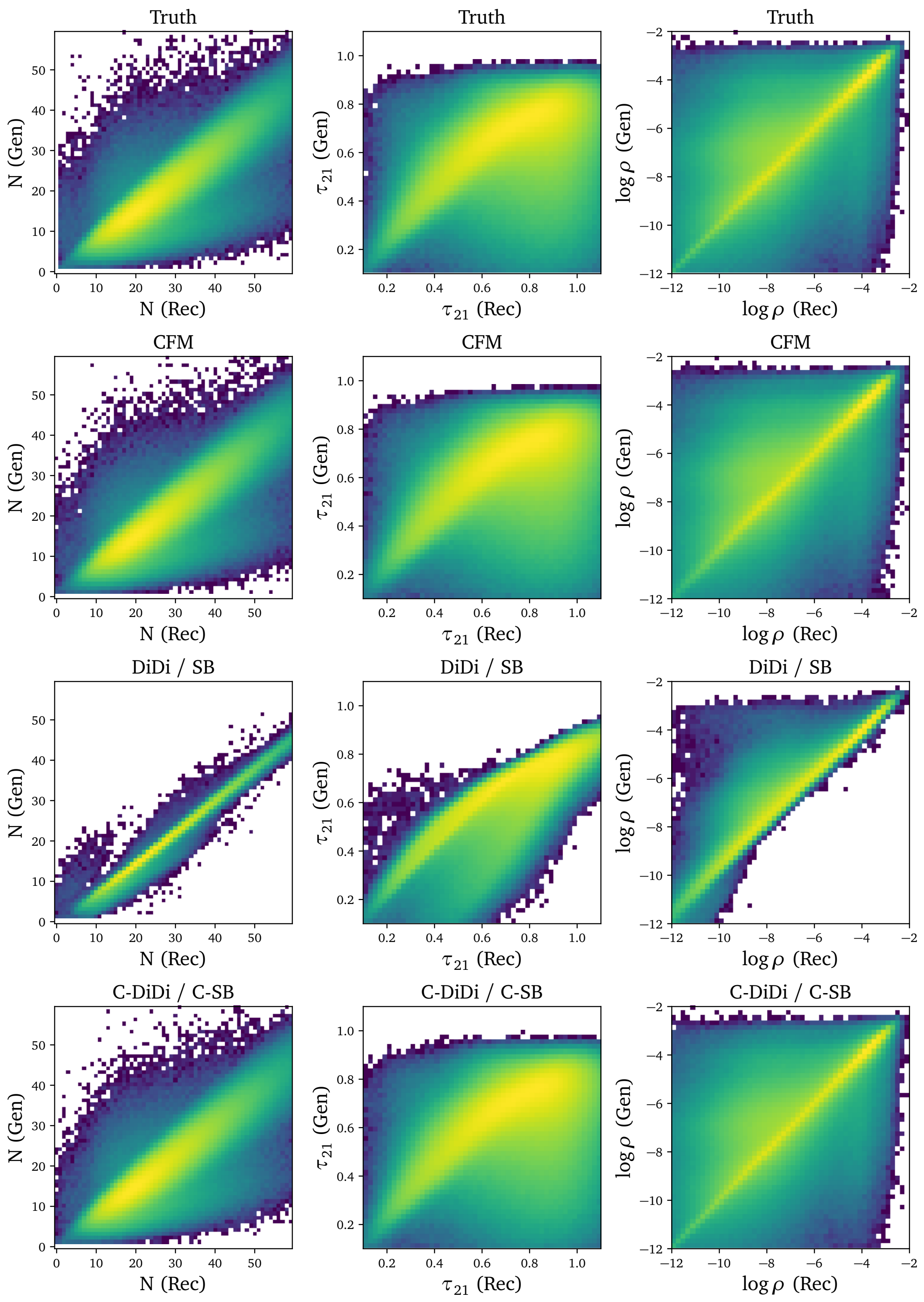


Figure 5.18: Migration maps for the 6D OmniFold dataset, truth compared to three different methods. We only show DiDi and C-DiDi, after verifying that the results are indistinguishable from SB and C-SB.

In Fig. 5.19, we show some single-event posterior distributions from the three methods,

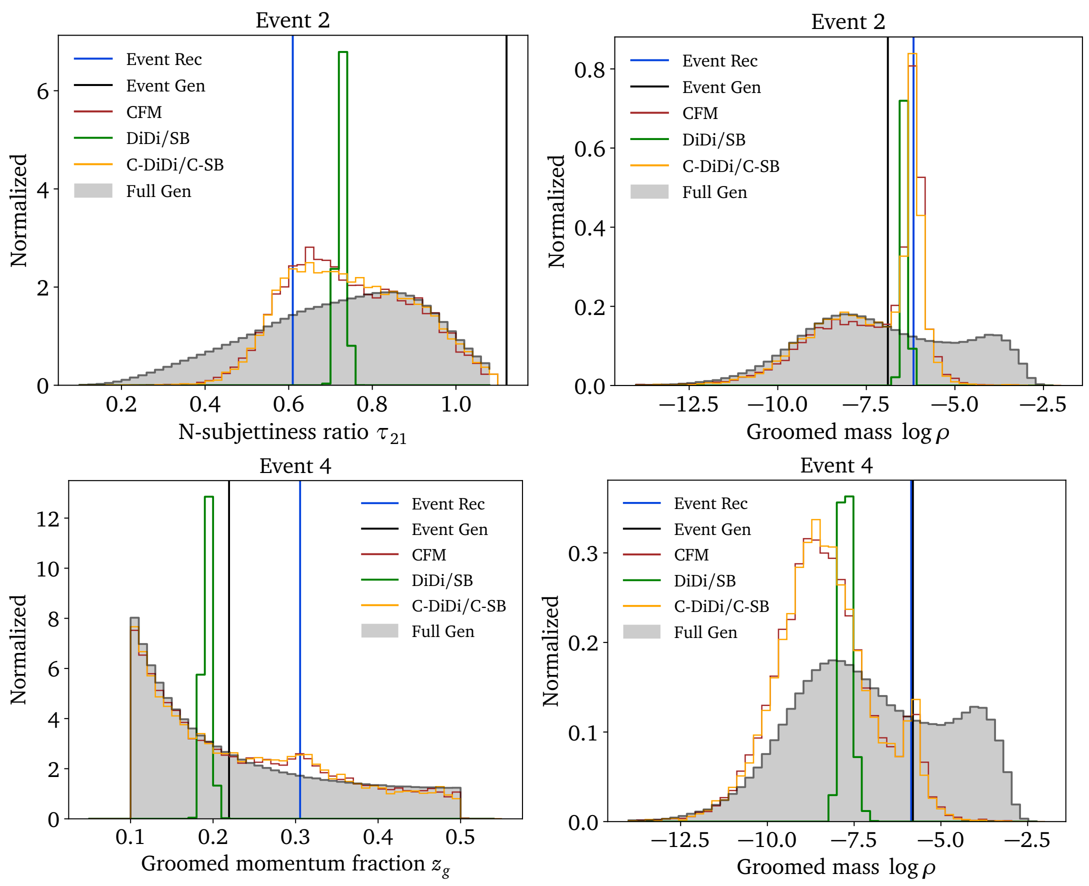


Figure 5.19: Posterior distributions obtained from unfolding single events with CFM, DiDi and C-DiDi on the 6D OmniFold dataset. Each of the three events is unfolded 10000 times. For reference, we also show the full gen-level distribution.

obtained by unfolding the same reco-event 10000 times. For reference, we include the unfolded event at reco-level, the gen-level truth, and the full unconditional gen-level distribution. Again, we observe a different behavior between the unconditional DM on the one hand and the CFM and conditional DM on the other. As expected from the derivation and from the migration plots, the unconditional DiDi network does not learn a physics-defined posterior, and our sampled distribution shows simply Gaussian smearing. The width of the Gaussian is related to the noise scale g of the SDE, which we verified by varying the noise scale over four orders of magnitude. The two conditional methods learn physically meaningful posteriors. Their shapes vary widely for the shown events and observables, but they agree between the different methods. We checked that the C-DiDi posteriors are invariant when varying the SDE noise scale g, so the learned single-event unfoldings illustrate how the CFM and the conditional DM-methods learn the same non-trivial conditional posteriors.

Classifier test

One approach to quantitatively test the performance of unfolding across the entire measured phase space is to use a post-hoc classifier, assuming that supervised classifier

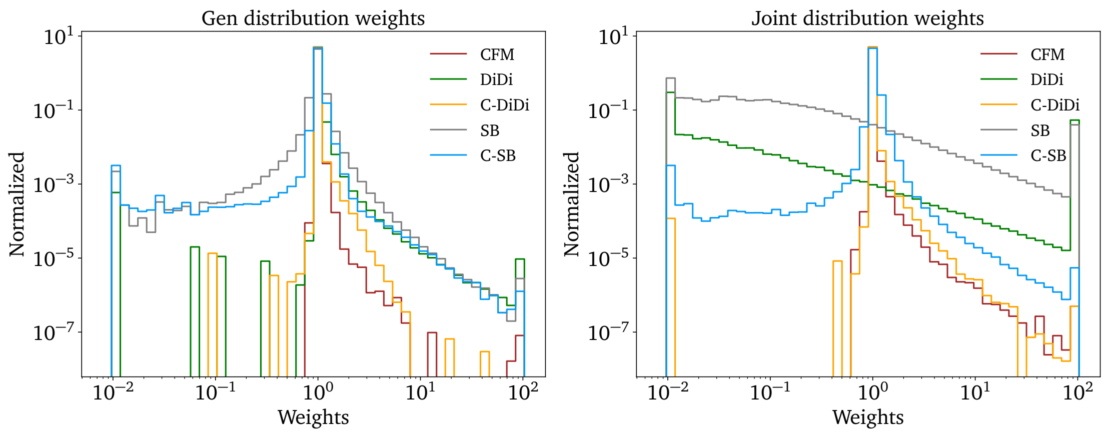


Figure 5.20: Classifier weight distributions for each network applied to the 6D OmniFold dataset. The left panel shows the gen-level weights according to Eq.(5.29), the right panel the joint distribution weights defined in Eq.(5.30).

training is more effective than unsupervised density estimation[3] as part of the generative networks [78, 92]. A well-trained and calibrated classifier C comparing training and generated events will approximate the likelihood ratio

$$w(x_\text{gen}) = \frac{p_\text{true}(x_\text{gen})}{p_\theta(x_\text{gen})} = \frac{C(x_\text{gen})}{1 - C(x_\text{gen})} \; . \tag{5.29}$$

With a slight modification, we can employ this technique to evaluate the quality of our learned posterior distributions. Instead of training the classifier only on gen-level, we train on the joint reco-level and gen-level data. This gives us access to the likelihood ratio of the joint distributions. Making use of the fact that the reco-level distribution is the same for generated and true, we can write

$$\begin{aligned} \frac{C(x_\text{gen}, x_\text{rec})}{1 - C(x_\text{gen}, x_\text{rec})} &= \frac{p_\text{true}(x_\text{gen}, x_\text{rec})}{p_\theta(x_\text{gen}, x_\text{rec})} \\ &= \frac{p_\text{true}(x_\text{rec}) p_\text{true}(x_\text{gen}|x_\text{rec})}{p_\theta(x_\text{rec}) p_\theta(x_\text{gen}|x_\text{rec})} \\ &= \frac{p_\text{true}(x_\text{gen}|x_\text{rec})}{p_\theta(x_\text{gen}|x_\text{rec})} \equiv w(x_\text{gen}|x_\text{rec}) \; . \end{aligned} \tag{5.30}$$

Therefore, a classifier trained on the joint distributions gives us access to the likelihood ratio between the true and learned posterior distributions.

For each of our five networks we train a classifier, using the hyperparameters listed in Tab. A.11. First we only look at the gen-level unnfolded distributions and discriminate them from the true gen-level distribution. We show the corresponding weight distributions, evaluated on the generated events, in the left panel of Fig. 5.20. For all networks we see a dominant peak in one, indicating that for the overwhelming majority of events, the classifier cannot tell truth from generated. The tail towards lower weights indicates events which should not be there and which the classifier weight tries to remove, i.e. phase space regions that the network overpopulates. The right tail marks events and phase space regions underpopulated by the generative unfolding network.

[3]In practice, if this is the case, the post-hoc classifier could be used to improve the quality of the generative model [171].

Comparing the five networks and the three underlying methods, the CFM shows the smallest tails in both directions. The conditional DiDi network is almost on par with the CFM, the difference is covered by the classifier training fluctuations. Unconditional DiDi leads to a larger tail towards large weights, but hardly any events with small weights, indicating a bias in learning features or their correlations. The SB networks show a slightly larger spread in the weight distribution, and the conditional SB is again significantly narrower than the unconditional version.

The right panel of Fig. 5.20 shows the weight distributions for the conditional phase space distributions. While for the conditional CFM and the conditional DM-networks the difference to the left panel is marginal, we now see that the unconditional DM-networks show little structure and large overflow bins, indicating that the learned joint distributions do not reproduce the training data.

5.2.3 Unfolding Substructure and Kinematic Properties

In order to stress-test the methods with a mixture of jet substructure and kinematic information, we simulated a dataset similar to the one used by a recent ATLAS analysis [161]. The resulting dataset has 22 instead of 6 dimensions, as in the previous dataset[4].

New $Z + 2$ jets dataset

We now consider the process

$$pp \to Z_{\mu\mu} + 2 \text{ jets}\,. \tag{5.31}$$

We generate the events with Madgraph 5 [23], showering and hadronization are simulated with Pythia 8.311 [172], and detector effects are included via Delphes 3.5.0 [28] using the default CMS card. Jets are clustered at gen-level and reco-level using an anti-k_T algorithm with $R = 0.4$ implemented in FastJet 3.3.4 [53].

We apply a set of selections resembling the ATLAS analysis: events are required to have exactly two muons with opposite charge and

$$p_{T,\mu} > 25 \text{ GeV} \qquad \text{and} \qquad m_{\mu\mu} \in [81, 101] \text{ GeV}\,. \tag{5.32}$$

Furthermore, we require at least two jets with

$$p_{T,j} > 10 \text{ GeV} \qquad \text{and} \qquad \Delta R_{\mu j} > 0.4\,. \tag{5.33}$$

All events must pass all selections on gen and reco level (acceptance effects are small and ignored). The training set consists of 1.5M events, and the test set consists of 400k events.

Instead of restricting ourselves to jet substructure observables of the leading jet, we now unfold both kinematic information of the muons and leading jets as well as the substructure of the leading two jets. At the subjet level, we include the number of jet

[4]The OmniFold dataset does have the full set of jet constituents, but this is a high-dimensional variable-length set, which is beyond the scope of this study.

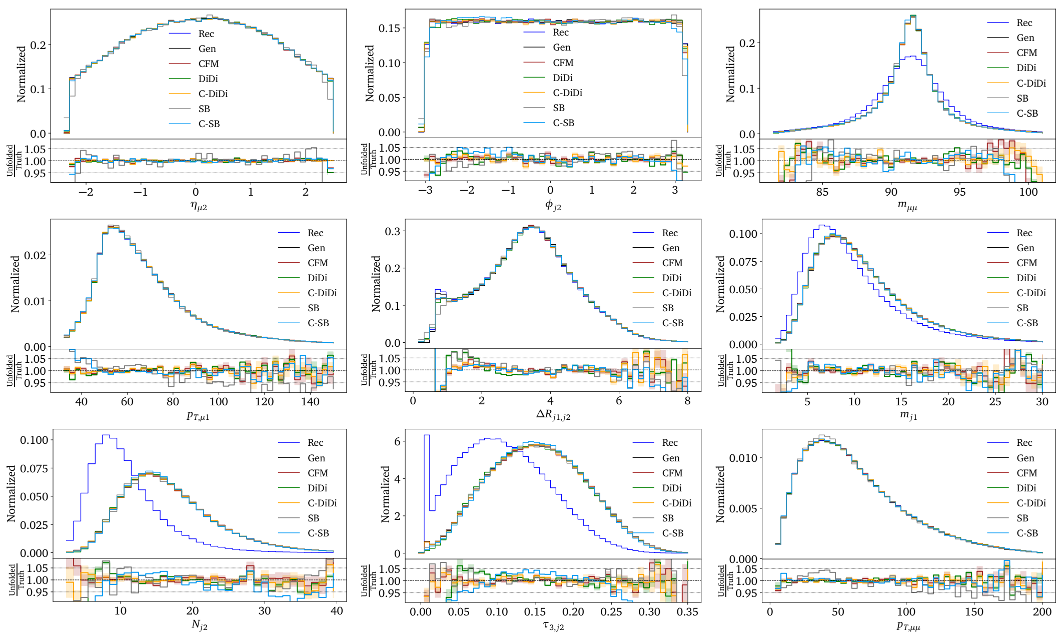


Figure 5.21: Unfolded distributions of the 22-dimensional Z+2 jets dataset using CFM, DiDi, C-DiDi, SB and C-SB.

constituents N and the subjettiness variables τ_1, τ_2 and τ_3 [167] for each jets. In total this defines a 22-dimensional phase space to unfold into:

$$\Big((p_T, \eta, \phi)_{\mu_1,\mu_2}\,,\; (p_T, \eta, \phi, m, N, \tau_1, \tau_2, \tau_3)_{j_1,j_2} \Big)\;. \tag{5.34}$$

The challenge is to reproduce all correlations, most notably the di-muon kinematics and the angular separation R_{jj}. In contrast to classifier-based unfolding, generative unfolding does not easily allow to over-constrain the physical phase space with redundant degrees of freedom. Instead, we choose a phase space parametrization that makes key correlations directly accessible to the network [78, 154]. To this end, we replace $p_{T,\mu_1} \to m_{\mu\mu}$ in the event representation and extract it later via

$$p_{T,\mu 1} = \frac{m_{\mu\mu}^2}{2p_{T,\mu_2}(\cosh \Delta\eta_{\mu\mu} - \cos \Delta\phi_{\mu\mu})}\,. \tag{5.35}$$

We standardize our training data in each dimension. For $m_{\mu\mu}$ we apply a Breit-Wigner mapping [78].

To accommodate the more challenging phase space, we replace the MLPs in all our generators with transformers [37,78,157]. We follow the transformer architecture proposed in Ref. [78], the network and training hyperparameters are listed in Tab. A.12 in the Appendix.

Unfolded distributions

The unfolding results obtained with all five networks representing all three methods are shown in Fig. 5.21. All of them unfold the bulk of the phase space distributions with a precision that is at the per-cent level. Small deviations from the true gen-level distributions as well as differences between different methods are only visible in the tails of the distributions or at hard edges. For $\eta_{\mu 2}$ and ϕ_{j2}, we expect the unfolding to be close to an identity mapping, with hard boundaries. While the DM-networks should be well-suited to those observables, we also find no performance loss for the classic CFM unfolding. Instead, we observe minor deviations of the (C-)SB networks at the hard edges, likely indicating a lack of expressivity in our specific implementation.

Since we turned it into a network input, the di-muon mass $m_{\mu\mu}$ is learned well by all networks. Moving on to complex correlations like the transverse momentum of the first muon, computed from Eq.(5.35), we still observe excellent agreement. With the exception of (C-)SB networks in the low-p_T region, the precision is at the level of a few per-cent except for fluctuating tails. A known challenge to all generative networks is the ΔR_{j1j2} distribution. This observable defines a non-trivial derived feature, combining collinear enhancement with a hard phase cut. All five networks struggle with this feature, and all conditional networks show superior performance.

At the subjet level, the number of jet constituents, N_{j2}, tends to be larger at gen-level than at reco-level, since not all particle are eventually detected. This explains the strange peak at zero for τ_{3,j_2} at reco-level. Here, the jet algorithm clusters less than two particles within one jet, indeed giving $\tau_{3,j_2} = 0$. At gen-level this effect is highly suppressed. The CFM and C-DiDi manage to reliably unfold even this small excess of zeros, while the other networks fit through it. DiDi and the SB deviate above 10% from the truth when getting closer to the tails of the distributions. To compensate the SB is overpopulating the peak region.

With the release of the updated methods, we have added a new ML-based unfolding method to the collider physics toolkit. Just as with classical unfolding, it is critical to have multiple, comparably accurate/precise techniques that have different methodological assumptions. Distribution mapping is a third type of method that can now be used to compare with standard conditional-generation and likelihood-ratio methods. Further work is required to fully integrate all aspects of a cross section measurement into generative unfolding (e.g. acceptance effects), but the core component (inverting the forward model) is now highly advanced. We look forward to the application of these methods to experimental data in the near future.

CHAPTER **6**

Conclusion and Outlook

Entering the High-Luminosity-LHC (HL-LHC) era will come with new and exciting precision measurements. To get there we still need to solve many computational and conceptional problems. LHC simulations are costly, and they are expected to exceed the available computational budget needed to meet the statistical demands of the HL-LHC. Additionally, a vast amount of data will be collected that must be thoroughly analyzed. New and robust analysis techniques need to be developed to handle high-statistics, high-dimensional data, enabling precise measurements of Standard Model parameters and potentially even the observation of physics beyond the Standard Model.

In this thesis, numerous options to incorporate generative machine learning into the LHC simulation chain, both in the forward and backward directions, were explored. We started with an overview of the relevant physical and computational concepts, with a particular focus on the LHC simulation chain on one hand and generative diffusion networks on the other.

We then studied the use of generative models for fast and precise augmentation of existing simulations. In particular, uncertainty-aware diffusion networks were developed, tested and applied to augment Z+jets simulations and simulations of leptonic $t\bar{t}$-decays including full off-shell effects. Furthermore, by combining the flexibility of diffusion models with the expressivity of the attention mechanism, we were able to generate precise and accurate calorimeter showers in high-dimensional phase spaces.

We reached state-of-the-art precision for all of the applications presented. For further improvements and a better understanding, some aspects still need to be considered before ML-based fast event generators can be reliably used in downstream LHC tasks. Uncertainties must be studied more rigorously and checked for proper coverage and calibration. Additionally, the finite size of training dataset needs to be taken into account. Previous studies [173, 174] have shown that even though training statistics are a limiting factor we can still reach a statistical amplification effect through ML-based data augmentation. While these aspects were not addressed within the context of this thesis, they remain active research questions, and we may hope to gain a deeper understanding of them in the coming years.

In the final part of this thesis, we explored generative unfolding techniques to correct observed data for detector effects. To this end, we made use of the pairing between detector-level and particle-level events in simulation and trained a generative diffusion model to generate samples from the posterior. In the first use case, the unfolding of detector effects in hadronic top-quark decays to measure the top-quark mass, we observed that, by construction, this posterior is dependent on a prior, i.e. on the particle-level distribution in MC. However, we were able to mitigate the prior dependence by broadening the prior distribution during training and utilizing global information of detector-level observables into our pipeline. This allowed us to drastically reduce the leading systemic uncertainty [137] of such a measurement. There are still some open questions to be considered. Background, efficiency and acceptance effects still need to be accounted for and all systematic uncertainties need to be propagated through the unfolding. Going beyond a purely phenomenological proof-of-concept for the first time, this study represents a major step to incorporate generative unfolding into existing LHC analyses, with the potential to greatly enhance precision measurements.
Finally, we also studied the development of new generative unfolding approaches based on distribution mapping algorithms. Although these algorithms are conceptually similar to the generative diffusion networks used earlier, they map directly between two physical distributions rather than from a Gaussian to a target distribution. In the context of unfolding, this allowed us to map detector-level events directly to particle-level events. By construction, these distribution mapping techniques do not recover the correct posterior distribution. However, we managed to augment the mapping such that they encode the correct posterior. While conceptually promising, in the physical applications we studied, the standard generative unfolding benchmarks reached a similar level of performance. This was an interesting study that added another valuable ML-based unfolding technique to the toolkit of unfolding algorithms. It could be that for a given, unexplored dataset, generative unfolding with distribution mapping may outperform existing unfolding pipelines.
In addition to the physical implications of our studies, we also continually updated the generative models and their architectures to follow recent developments in the machine learning literature. Like current state-of-the-art image generators, we primarily employed generative diffusion networks with transformer backbones for all presented results. However, as machine learning continues to evolve rapidly, so do we. Most results in this thesis do not rely on a specific generative architecture, which means future models can easily replace the ones used here.

In conclusion, generative machine learning has matured to the point where it can be used with high precision and reliability in collider physics. While further validation and development are needed before full integration into experimental workflows, the pace of innovation in both high-energy physics and machine learning suggests that ML-based algorithms will soon become a standard part of the LHC simulation and analysis toolbox.

APPENDIX **A**

Hyperparameters

All hyperparameters used in the context of the thesis are listed in this section.

Fast Event Generation

hyperparameter	toy models	LHC events
Timesteps	1000	1000
Time Embedding Dimension	-	64
# Blocks	1	2
Layers per Block	8	5
Intermediate Dimensions	40	64
# Model Parameters	20k	75k
LR Scheduling	one-cycle	one-cycle
Starter LR	10^{-4}	10^{-4}
Maximum LR	10^{-3}	10^{-3}
Epochs	1000	1000, 3000, 10000
Batch Size	8192	8192, 8192, 4096
# Training Events	600k	3.2M, 850k, 190k
# Generated Events	1M	1M, 1M, 1M

Table A.1: Training setup and hyperparameters for the Bayesian DDPM generator introduced in Section 4.1 and taken from Ref. [1].

Hyperparameter	
Embedding dimension	64
Layers	8
Intermediate dimensions	768
LR scheduling	OneCycle
Starter LR	10^{-4}
Max LR	10^{-3}
Epochs	1000
Batch size	16384
c	10^{-3}
# Training events	3 M

Table A.2: Generative network setup (DiDi) and hyperparameters for all results discussed in Section 4.2 and taken from Ref. [2].

Hyperparameter	
Layers	5
Intermediate dimensions	512
Dropout	0.1
Normalization	BatchNorm1d
LR scheduling	ReduceOnPlateau
Starter LR	1^{-3}
Patience	10
Epochs	100
Batch size	1024
# Training events	2.5 M

Table A.3: Classifier network setup and hyperparameters for evaluating DiDi in Section 4.2 and taken from Ref. [2].

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 A.4: Parameters for the autoregressive energy network in Section 4.3.1, taken from Ref. [3]

	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 A.5: Parameters for the shape networks in Section 4.3.1, for the full and the latent space, taken from Ref. [3]

Parameter	Value
Optimizer	Adam
Learning rate	$2 \cdot 10^{-4}$
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 A.6: Parameters for HL and LL classifiers network used to calculate the weights of Fig. 4.20 taken from Ref. [3]. The other classifiers use the same hyperparameters but without any dropout.

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 A.7: Parameters of the autoencoder for DS2 and DS3 used for the laViT network in Section 4.3.1, taken from Ref. [3].

Generative Unfolding

Parameter	4D	6D
Epochs	800	1500
LR sched.	cosine	cosine
Max LR	10^{-3}	10^{-3}
Optimizer	Adam	Adam
Batch size	16384	16384
Network	Resnet	Resnet
Dim embedding	64	64
Intermediate dim	512	256
Num layers	8	8

Table A.8: Parameters for the 4-dimensional and 6-dimensional networks in Section 5.1.3.

Parameter	4D	6D
Epochs	800	500(+1000)
LR sched.	cosine	cosine
Max LR	10^{-3}	10^{-3}
Optimizer	Adam	Adam
Batch size	16384	16384
Dropout	0.1	0.1
Network	Transfusion	Transfusion
Dim embedding	64	64
Intermediate dim	512	512
Num layers	4	4
Num heads	4	4

Table A.9: Parameters for the 4-dimensional and 6-dimensional networks in Section 5.1.4.

Parameter	12D
Epochs	500
LR sched.	cosine
Max LR	10^{-3}
Optimizer	Adam
Batch size	16384
Dropout	0.1
Network	Transfusion
Dim embedding	128
Intermediate dim	512
Num layers	6
Num heads	4

Table A.10: Parameters for the 12-dimensional network in Section 5.1.6.

Parameter	CFM	DiDi	C-DiDi	SB	C-SB	Classifier
Optimizer		Adam		Adam		Adam
Learning rate		0.001		0.001		0.001
LR schedule		Cosine annealing		Exponential decay		Cosine annealing
Batch size		16384		128		128
Epochs		300		20		50
Network		MLP		MLP		MLP
Number of layers		5		6		5
Hidden nodes		128		256		256
Dropout		-		-		0.1
Noise scale	-	0.1	0.1	0.1	0.1	-
BNN regularization		1		-		-

Table A.11: Network and training hyperparameters for all networks trained to unfold the 6d jet substructure dataset in Section 5.2.2 taken from Ref. [4].

Parameter	CFM	DiDi	C-DiDi	SB	C-SB
Optimizer		Adam		Adam	
Learning rate		0.001		0.001	
LR schedule		Cosine annealing		Exponential decay	
Batch size		16384		128	
Epochs	500	500	2000	200	200
Network		Transformer		Transformer	
Embedding dim		64		64	
Transformer blocks		6		6	
Attention heads		4		4	
Feedforward dim		256		256	
Noise scale	-	0.001	0.1	0.1	0.1
BNN regularization		1		-	

Table A.12: Network and training hyperparameters for all networks trained to unfold the full-dimensional Z+2j dataset of Section 5.2.3 taken from Ref. [4].

Acknowledgments

I would like to thank Anja Butter for taking me on as a PhD student and for the endless travel opportunities. I also thank Björn Malte Schäfer for co-referring this thesis. Furthermore, I thank Stephanie Hansmann-Menzemer for being part of my committee and Tilman Plehn for not only completing my committee, but also for always having my back in the last two and a half years and for pretending that my talks are convincing.
The work presented in this thesis could not have been done without all my collaborators Nathan Hütsch, Luigi Favaro, Jonas Spinner, Ayodore Ore, Dennis Schwarz, Roman Kogler, Alexander Paasch, Peter Sorrenson, Vinicius Mikuni, Benjamin Nachman, Sascha Diefenbacher, Mathias Kuschick, Tomáš Ježo, Michael Klasen, Anja Butter and of course Tilman Plehn.
I also thank Henning Bahl, Theo Heimel, Ramon Winterhalder and Nikita Schmal for their support with handling MadGraph. A very special thank you to Henning Bahl, Ayodore Ore, Nikita Schmal and Nathan Hütsch for correcting this thesis.
Moving on to the Heidelberg Pheno and Cosmo group, I would like to thank: Nathan Hütsch for absolutely everything, Luigi Favaro for his infinite calm and patience, Benedikt Schosser and Lennart Röver for allowing me to snack and reset (and occasionally nap) in their Cosmo office, Jonas Spinner for racing through the rain with me at TASI, Giovanni De Crescenzo and Nikita Schmal for thinking that I am funny, Javier Mariño for reminding me that I am usually not, Rebecca Kuntz for indulging in nostalgia with me and Theo Heimel and Henning Bahl for their wisdom. In no particular order I would also like to thank Ayodore Ore, Claudius Krause, Maeve Madigan, Michael Luchmann, Paula Schuchard, Lorenz Vogel, Nina Elma, Victor Breso Pla and Emma Geoffray.
Lastly, I thank my friends, especially Malin and Vincent, and my family for their constant support.

Bibliography

[1] A. Butter, N. Huetsch, S. Palacios Schweitzer, T. Plehn, P. Sorrenson and J. Spinner, *Jet diffusion versus JetGPT – Modern networks for the LHC*, SciPost Phys. Core **8**, 026 (2025), doi:10.21468/SciPostPhysCore.8.1.026, arXiv:2305.10475.

[2] A. Butter, T. Jezo, M. Klasen, M. Kuschick, S. Palacios Schweitzer and T. Plehn, *Kicking it off(-shell) with direct diffusion*, SciPost Phys. Core **7**, 064 (2024), doi:10.21468/SciPostPhysCore.7.3.064, arXiv:2311.17175.

[3] L. Favaro, A. Ore, S. Palacios Schweitzer and T. Plehn, *CaloDREAM – Detector Response Emulation via Attentive flow Matching*, SciPost Phys. **18**, 088 (2025), doi:10.21468/SciPostPhys.18.3.088, arXiv:2405.09629.

[4] A. Butter, S. Diefenbacher, N. Huetsch, V. Mikuni, B. Nachman, S. Palacios Schweitzer and T. Plehn, *Generative Unfolding with Distribution Mapping* (2024), arXiv:2411.02495.

[5] L. Favaro, R. Kogler, A. Paasch, S. Palacios Schweitzer, T. Plehn and D. Schwarz, *How to Unfold Top Decays* (2025), arXiv:2501.12363.

[6] O. Amram *et al.*, *CaloChallenge 2022: A Community Challenge for Fast Calorimeter Simulation* (2024), arXiv:2410.21611.

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

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

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

[10] G. 't Hooft and M. J. G. Veltman, *Regularization and Renormalization of Gauge Fields*, Nucl. Phys. B **44**, 189 (1972), doi:10.1016/0550-3213(72)90279-9.

[11] H. Fritzsch, M. Gell-Mann and H. Leutwyler, *Advantages of the Color Octet Gluon Picture*, Phys. Lett. B **47**, 365 (1973), doi:10.1016/0370-2693(73)90625-4.

[12] D. J. Gross and F. Wilczek, *Ultraviolet Behavior of Non-Abelian Gauge Theories*, Phys. Rev. Lett. **30**, 1343 (1973), doi:10.1103/PhysRevLett.30.1343.

[13] H. D. Politzer, *Reliable Perturbative Results for Strong Interactions?*, Phys. Rev. Lett. **30**, 1346 (1973), doi:10.1103/PhysRevLett.30.1346.

[14] 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.

[15] 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.

[16] P. A. R. Ade *et al.*, *Planck 2015 results. XIII. Cosmological parameters*, Astron. Astrophys. **594**, A13 (2016), doi:10.1051/0004-6361/201525830, arXiv:1502.01589.

[17] R. Adam *et al.*, *Planck 2015 results. I. Overview of products and scientific results*, Astron. Astrophys. **594**, A1 (2016), doi:10.1051/0004-6361/201527101, arXiv:1502.01582.

[18] E. W. Kolb and M. S. Turner, *The Early Universe*, vol. 69, Taylor and Francis, ISBN 978-0-429-49286-0, 978-0-201-62674-2, doi:10.1201/9780429492860 (2019).

[19] G. Steigman, *Observational tests of antimatter cosmologies*, Ann. Rev. Astron. Astrophys. **14**, 339 (1976), doi:10.1146/annurev.aa.14.090176.002011.

[20] J. E. Kim and G. Carosi, *Axions and the Strong CP Problem*, Rev. Mod. Phys. **82**, 557 (2010), doi:10.1103/RevModPhys.82.557, [Erratum: Rev.Mod.Phys. 91, 049902 (2019)], arXiv:0807.3125.

[21] Y. Fukuda *et al.*, *Evidence for oscillation of atmospheric neutrinos*, Phys. Rev. Lett. **81**, 1562 (1998), doi:10.1103/PhysRevLett.81.1562, arXiv:hep-ex/9807003.

[22] Q. R. Ahmad *et al.*, *Direct evidence for neutrino flavor transformation from neutral current interactions in the Sudbury Neutrino Observatory*, Phys. Rev. Lett. **89**, 011301 (2002), doi:10.1103/PhysRevLett.89.011301, arXiv:nucl-ex/0204008.

[23] 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.

[24] 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.

[25] 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.

[26] M. Bahr *et al.*, *Herwig++ Physics and Manual*, Eur. Phys. J. C **58**, 639 (2008), doi:10.1140/epjc/s10052-008-0798-9, arXiv:0803.0883.

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

[28] 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.

[29] 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.

[30] 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.

[31] T. Heimel, N. Huetsch, F. Maltoni, O. Mattelaer, T. Plehn and R. Winterhalder, *The MadNIS reloaded*, SciPost Phys. **17**, 023 (2024), doi:10.21468/SciPostPhys.17.1.023, arXiv:2311.01548.

[32] T. Heimel, O. Mattelaer, T. Plehn and R. Winterhalder, *Differentiable MadNIS-Lite*, SciPost Phys. **18**, 017 (2025), doi:10.21468/SciPostPhys.18.1.017, arXiv:2408.01486.

[33] 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.

[34] 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.

[35] K. Cranmer, J. Brehmer and G. Louppe, *The frontier of simulation-based inference*, Proc. Nat. Acad. Sci. **117**, 30055 (2020), doi:10.1073/pnas.1912789117, arXiv:1911.01429.

[36] A. Butter, T. Heimel, T. Martini, S. Peitzsch and T. Plehn, *Two invertible networks for the matrix element method*, SciPost Phys. **15**, 094 (2023), doi:10.21468/SciPostPhys.15.3.094, arXiv:2210.00019.

[37] T. Heimel, N. Huetsch, R. Winterhalder, T. Plehn and A. Butter, *Precision-machine learning for the matrix element method*, SciPost Phys. **17**, 129 (2024), doi:10.21468/SciPostPhys.17.5.129, arXiv:2310.07752.

[38] V. Chekhovsky *et al.*, *Model-agnostic search for dijet resonances with anomalous jet substructure in proton-proton collisions at $\sqrt{s} = 13$ TeV* (2024), arXiv:2412.03747.

[39] J. Ho, A. Jain and P. Abbeel, *Denoising Diffusion Probabilistic Models* (2020), arXiv:2006.11239.

[40] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel and M. Le, *Flow matching for generative modeling*, arXiv preprint arXiv:2210.02747 (2022).

[41] 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.

[42] 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).

[43] F. Quevedo and A. Schachner, *Cambridge Lectures on The Standard Model* (2024), arXiv:2409.09211.

[44] G. Aad *et al.*, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3**, S08003 (2008).

[45] S. Chatrchyan *et al.*, *The CMS experiment at the CERN LHC*, JINST **3**, S08004 (2008).

[46] T. Plehn, *Lectures on LHC Physics*, doi:10.1007/978-3-319-05942-6 (2015).

[47] 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.

[48] J. Apostolakis *et al.*, *HEP Software Foundation Community White Paper Working Group - Detector Simulation* (2018), arXiv:1803.04165.

[49] S. D. Ellis and D. E. Soper, *Successive combination jet algorithm for hadron collisions*, Phys. Rev. D **48**, 3160 (1993), doi:10.1103/PhysRevD.48.3160, arXiv:hep-ph/9305266.

[50] 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.

[51] M. Wobisch and T. Wengler, *Hadronization corrections to jet cross-sections in deep inelastic scattering*, In *Workshop on Monte Carlo Generators for HERA Physics (Plenary Starting Meeting)*, pp. 270–279 (1998), arXiv:hep-ph/9907280.

[52] I. W. Stewart, F. J. Tackmann, J. Thaler, C. K. Vermilion and T. F. Wilkason, *XCone: N-jettiness as an Exclusive Cone Jet Algorithm*, JHEP **11**, 072 (2015), doi:10.1007/JHEP11(2015)072, arXiv:1508.01516.

[53] M. Cacciari, G. P. Salam and G. Soyez, *FastJet User Manual*, Eur. Phys. J. C **72**, 1896 (2012), doi:10.1140/epjc/s10052-012-1896-2, arXiv:1111.6097.

[54] M. Beneke *et al.*, *Top quark physics*, hep-ph/0003033 (2000), arXiv:hep-ph/0003033.

[55] S. Navas *et al.*, *Review of particle physics*, Phys. Rev. D **110**, 030001 (2024), doi:10.1103/PhysRevD.110.030001.

[56] A. M. Sirunyan *et al.*, *Measurement of the top quark mass in the dileptonic $t\bar{t}$ decay channel using the mass observables $M_{b\ell}$, M_{T2}, and $M_{b\ell\nu}$ in pp collisions at $\sqrt{s} = 8$ TeV*, Phys. Rev. D **96**, 032002 (2017), doi:10.1103/PhysRevD.96.032002, arXiv:1704.06142.

[57] A. H. Hoang, *The Top Mass: Interpretation and Theoretical Uncertainties*, In *7th International Workshop on Top Quark Physics* (2014), arXiv:1412.3649.

[58] A. H. Hoang, *What is the Top Quark Mass?*, Ann. Rev. Nucl. Part. Sci. **70**, 225 (2020), doi:10.1146/annurev-nucl-101918-023530, arXiv:2004.12915.

[59] A. H. Hoang, S. Mantry, A. Pathak and I. W. Stewart, *Extracting a Short Distance Top Mass with Light Grooming*, Phys. Rev. D **100**, 074021 (2019), doi:10.1103/PhysRevD.100.074021, arXiv:1708.02586.

[60] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization* (2017), arXiv:1412.6980.

[61] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15**, 1929 (2014).

[62] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, In *International Conference on Machine Learning (ICML)*, pp. 448–456 (2015).

[63] J. L. Ba, J. R. Kiros and G. E. Hinton, *Layer normalization*, arXiv preprint arXiv:1607.06450 (2016).

[64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention is all you need* (2023), arXiv:1706.03762.

[65] A. Butter, F. Charton, J. M. n. Villadamigo, A. Ore, T. Plehn and J. Spinner, *Extrapolating Jet Radiation with Autoregressive Transformers* (2024), arXiv:2412.12074.

[66] 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.

[67] D. MacKay, *Probable Networks and Plausible Predictions – A Review of Practical Bayesian Methods for Supervised Neural Networks*, Comp. in Neural Systems **6**, 4679 (1995), doi:10.1088/0954-898X/6/3/011.

[68] R. M. Neal, *Bayesian learning for neural networks*, Ph.D. thesis, Toronto (1995).

[69] Y. Gal, *Uncertainty in Deep Learning*, Ph.D. thesis, Cambridge (2016).

[70] D. M. Blei, A. Kucukelbir and J. D. McAuliffe, *Variational inference: A review for statisticians*, Journal of the American statistical Association **112**, 859 (2017), doi:doi:10.1080/01621459.2017.1285773.

[71] G. Kasieczka, M. Luchmann, F. Otterpohl and T. Plehn, *Per-Object Systematics using Deep-Learned Calibration*, SciPost Phys. **9**, 089 (2020), doi:10.21468/SciPostPhys.9.6.089, arXiv:2003.11099.

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

[73] D. P. Kingma, T. Salimans, B. Poole and J. Ho, *Variational diffusion models*, doi:10.48550/ARXIV.2107.00630 (2021).

[74] X. Liu, C. Gong and Q. Liu, *Flow straight and fast: Learning to generate and transfer data with rectified flow* (2022), arXiv:2209.03003.

[75] M. S. Albergo and E. Vanden-Eijnden, *Building normalizing flows with stochastic interpolants* (2023), arXiv:2209.15571.

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

[77] S. Diefenbacher, G.-H. Liu, V. Mikuni, B. Nachman and W. Nie, *Improving generative model-based unfolding with Schrödinger bridges*, Phys. Rev. D **109**, 076011 (2024), doi:10.1103/PhysRevD.109.076011, arXiv:2308.12351.

[78] N. Huetsch *et al.*, *The landscape of unfolding with machine learning*, SciPost Phys. **18**, 070 (2025), doi:10.21468/SciPostPhys.18.2.070, arXiv:2404.18807.

[79] B. D. Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications **12**, 313 (1982).

[80] J. Doob, *Conditional brownian motion and the boundary limits of harmonic functions*, Bulletin de la Société Mathématique de France **85**, 431 (1957).

[81] G.-H. Liu, A. Vahdat, D.-A. Huang, E. A. Theodorou, W. Nie and A. Anandkumar, *I^2sb: Image-to-image schrödinger bridge* (2023), arXiv:2302.05872.

[82] V. D. Bortoli, G.-H. Liu, T. Chen, E. A. Theodorou and W. Nie, *Augmented bridge matching* (2023), arXiv:2311.06978.

[83] 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`.

[84] A. Paszke *et al.*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc. (2019), arXiv:1912.01703.

[85] P. Ramachandran, B. Zoph and Q. V. Le, *Swish: a self-gated activation function*, arXiv preprint arXiv:1710.05941 (2017).

[86] S. Catani, F. Krauss, R. Kuhn and B. R. Webber, *QCD matrix elements + parton showers*, JHEP **11**, 063 (2001), doi:10.1088/1126-6708/2001/11/063, arXiv:hep-ph/0109231.

[87] 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.

[88] 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.

[89] G. Heinrich, A. Maier, R. Nisius, J. Schlenk, M. Schulze, L. Scyboz and J. Winter, *NLO and off-shell effects in top quark mass determinations*, JHEP **07**, 129 (2018), doi:10.1007/JHEP07(2018)129, arXiv:1709.08615.

[90] S. Ferrario Ravasio, T. Ježo, P. Nason and C. Oleari, *A theoretical study of top-mass measurements at the LHC using NLO+PS generators of increasing accuracy*, Eur. Phys. J. C **78**, 458 (2018), doi:10.1140/epjc/s10052-019-7336-9, [Addendum: Eur.Phys.J.C 79, 859 (2019)], arXiv:1906.09166.

[91] T. Golling, S. Klein, R. Mastandrea, B. Nachman and J. A. Raine, *Morphing one dataset into another with maximum likelihood estimation*, Phys. Rev. D **108**, 096018 (2023), doi:10.1103/PhysRevD.108.096018, arXiv:2309.06472.

[92] 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.

[93] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC Events*, SciPost Phys. **7**, 075 (2019), doi:10.21468/SciPostPhys.7.6.075, arXiv:1907.03764.

[94] M. Czakon, P. Fiedler and A. Mitov, *Total Top-Quark Pair-Production Cross Section at Hadron Colliders Through $O(\alpha_S^4)$*, Phys. Rev. Lett. **110**, 252004 (2013), doi:10.1103/PhysRevLett.110.252004, arXiv:1303.6254.

[95] M. Czakon, D. Heymes and A. Mitov, *High-precision differential predictions for top-quark pairs at the LHC*, Phys. Rev. Lett. **116**, 082003 (2016), doi:10.1103/PhysRevLett.116.082003, arXiv:1511.00549.

[96] S. Catani, S. Devoto, M. Grazzini, S. Kallweit, J. Mazzitelli and H. Sargsyan, *Top-quark pair hadroproduction at next-to-next-to-leading order in QCD*, Phys. Rev. D **99**, 051501 (2019), doi:10.1103/PhysRevD.99.051501, arXiv:1901.04005.

[97] S. Catani, S. Devoto, M. Grazzini, S. Kallweit and J. Mazzitelli, *Top-quark pair production at the LHC: Fully differential QCD predictions at NNLO*, JHEP **07**, 100 (2019), doi:10.1007/JHEP07(2019)100, arXiv:1906.06535.

[98] W. Bernreuther, M. Fucker and Z.-G. Si, *Weak interaction corrections to hadronic top quark pair production: Contributions from quark-gluon and b anti-b induced reactions*, Phys. Rev. D **78**, 017503 (2008), doi:10.1103/PhysRevD.78.017503, arXiv:0804.1237.

[99] J. H. Kuhn, A. Scharf and P. Uwer, *Electroweak effects in top-quark pair production at hadron colliders*, Eur. Phys. J. C **51**, 37 (2007), doi:10.1140/epjc/s10052-007-0275-x, arXiv:hep-ph/0610335.

[100] W. Hollik and D. Pagani, *The electroweak contribution to the top quark forward-backward asymmetry at the Tevatron*, Phys. Rev. D **84**, 093003 (2011), doi:10.1103/PhysRevD.84.093003, arXiv:1107.2606.

[101] C. Gütschow, J. M. Lindert and M. Schönherr, *Multi-jet merged top-pair production including electroweak corrections*, Eur. Phys. J. C **78**, 317 (2018), doi:10.1140/epjc/s10052-018-5804-2, arXiv:1803.00950.

[102] R. Frederix, I. Tsinikos and T. Vitos, *Probing the spin correlations of $t\bar{t}$ production at NLO QCD+EW*, Eur. Phys. J. C **81**, 817 (2021), doi:10.1140/epjc/s10052-021-09612-9, arXiv:2105.11478.

[103] M. Czakon, D. Heymes, A. Mitov, D. Pagani, I. Tsinikos and M. Zaro, *Top-pair production at the LHC through NNLO QCD and NLO EW*, JHEP **10**, 186 (2017), doi:10.1007/JHEP10(2017)186, arXiv:1705.04105.

[104] J. Gao, C. S. Li and H. X. Zhu, *Top Quark Decay at Next-to-Next-to Leading Order in QCD*, Phys. Rev. Lett. **110**, 042001 (2013), doi:10.1103/PhysRevLett.110.042001, arXiv:1210.2808.

[105] M. Brucherseifer, F. Caola and K. Melnikov, *$\mathcal{O}(\alpha_s^2)$ corrections to fully-differential top quark decays*, JHEP **04**, 059 (2013), doi:10.1007/JHEP04(2013)059, arXiv:1301.7133.

[106] J. Gao and A. S. Papanastasiou, *Top-quark pair-production and decay at high precision*, Phys. Rev. D **96**, 051501 (2017), doi:10.1103/PhysRevD.96.051501, arXiv:1705.08903.

[107] A. Behring, M. Czakon, A. Mitov, A. S. Papanastasiou and R. Poncelet, *Higher order corrections to spin correlations in top quark pair production at the LHC*, Phys. Rev. Lett. **123**, 082001 (2019), doi:10.1103/PhysRevLett.123.082001, arXiv:1901.05407.

[108] M. Czakon, A. Mitov and R. Poncelet, *NNLO QCD corrections to leptonic observables in top-quark pair production and decay*, JHEP **05**, 212 (2021), doi:10.1007/JHEP05(2021)212, arXiv:2008.11133.

[109] G. Bevilacqua, M. Czakon, A. van Hameren, C. G. Papadopoulos and M. Worek, *Complete off-shell effects in top quark pair hadroproduction with leptonic decay at next-to-leading order*, JHEP **02**, 083 (2011), doi:10.1007/JHEP02(2011)083, arXiv:1012.4230.

[110] A. Denner, S. Dittmaier, S. Kallweit and S. Pozzorini, *NLO QCD corrections to WWbb production at hadron colliders*, Phys. Rev. Lett. **106**, 052001 (2011), doi:10.1103/PhysRevLett.106.052001, arXiv:1012.3975.

[111] A. Denner, S. Dittmaier, S. Kallweit and S. Pozzorini, *NLO QCD corrections to off-shell top-antitop production with leptonic decays at hadron colliders*, JHEP **10**, 110 (2012), doi:10.1007/JHEP10(2012)110, arXiv:1207.5018.

[112] G. Heinrich, A. Maier, R. Nisius, J. Schlenk and J. Winter, *NLO QCD corrections to $W^+W^-b\bar{b}$ production with leptonic decays in the light of top quark mass and asymmetry measurements*, JHEP **06**, 158 (2014), doi:10.1007/JHEP06(2014)158, arXiv:1312.6659.

[113] R. Frederix, *Top Quark Induced Backgrounds to Higgs Production in the $WW^{(*)} \to ll\nu\nu$ Decay Channel at Next-to-Leading-Order in QCD*, Phys. Rev. Lett. **112**, 082002 (2014), doi:10.1103/PhysRevLett.112.082002, arXiv:1311.4893.

[114] F. Cascioli, S. Kallweit, P. Maierhöfer and S. Pozzorini, *A unified NLO description of top-pair and associated Wt production*, Eur. Phys. J. C **74**, 2783 (2014), doi:10.1140/epjc/s10052-014-2783-9, arXiv:1312.0546.

[115] G. Bevilacqua, H. B. Hartanto, M. Kraus and M. Worek, *Top Quark Pair Production in Association with a Jet with Next-to-Leading-Order QCD Off-Shell Effects at the Large Hadron Collider*, Phys. Rev. Lett. **116**, 052003 (2016), doi:10.1103/PhysRevLett.116.052003, arXiv:1509.09242.

[116] R. D. Ball *et al.*, *Parton distributions from high-precision collider data*, Eur. Phys. J. C **77**, 663 (2017), doi:10.1140/epjc/s10052-017-5199-5, arXiv:1706.00428.

[117] S. Frixione, P. Nason and G. Ridolfi, *A Positive-weight next-to-leading-order Monte Carlo for heavy flavour hadroproduction*, JHEP **09**, 126 (2007), doi:10.1088/1126-6708/2007/09/126, arXiv:0707.3088.

[118] T. Ježo, J. M. Lindert, P. Nason, C. Oleari and S. Pozzorini, *An NLO+PS generator for $t\bar{t}$ and Wt production and decay including non-resonant and interference effects*, Eur. Phys. J. C **76**, 691 (2016), doi:10.1140/epjc/s10052-016-4538-2, arXiv:1607.04538.

[119] T. Ježo, J. M. Lindert and S. Pozzorini, *Resonance-aware NLOPS matching for off-shell $t\bar{t}$ + tW production with semileptonic decays*, JHEP **10**, 008 (2023), doi:10.1007/JHEP10(2023)008, arXiv:2307.15653.

[120] S. Frixione, E. Laenen, P. Motylinski and B. R. Webber, *Angular correlations of lepton pairs from vector boson and top quark decays in Monte Carlo simulations*, JHEP **04**, 081 (2007), doi:10.1088/1126-6708/2007/04/081, arXiv:hep-ph/0702198.

[121] P. Nason, *A New method for combining NLO QCD with shower Monte Carlo algorithms*, JHEP **11**, 040 (2004), doi:10.1088/1126-6708/2004/11/040, arXiv:hep-ph/0409146.

[122] S. Frixione, P. Nason and C. Oleari, *Matching NLO QCD computations with Parton Shower simulations: the POWHEG method*, JHEP **11**, 070 (2007), doi:10.1088/1126-6708/2007/11/070, arXiv:0709.2092.

[123] S. Alioli, P. Nason, C. Oleari and E. Re, *A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX*, JHEP **06**, 043 (2010), doi:10.1007/JHEP06(2010)043, arXiv:1002.2581.

[124] T. Ježo and P. Nason, *On the Treatment of Resonances in Next-to-Leading Order Calculations Matched to a Parton Shower*, JHEP **12**, 065 (2015), doi:10.1007/JHEP12(2015)065, arXiv:1509.09071.

[125] A. Tong, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, K. Fatras, G. Wolf and Y. Bengio, *Improving and generalizing flow-based generative models with minibatch optimal transport* (2023), arXiv:2302.00482.

[126] F. Ernst, L. Favaro, C. Krause, T. Plehn and D. Shih, *Normalizing Flows for High-Dimensional Detector Simulations*, SciPost Phys. **18**, 081 (2025), doi:10.21468/SciPostPhys.18.3.081, arXiv:2312.09290.

[127] 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.

[128] 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.

[129] 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).

[130] 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).

[131] M. Faucci Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 github page*, `https://github.com/CaloChallenge/homepage` (2022).

[132] 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.

[133] 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.

[134] 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.

[135] 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.

[136] 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.

[137] A. Tumasyan *et al.*, *Measurement of the differential $t\bar{t}$ production cross section as a function of the jet mass and extraction of the top quark mass in hadronic decays of boosted top quarks*, Eur. Phys. J. C **83**, 560 (2023), doi:10.1140/epjc/s10052-023-11587-8, arXiv:2211.01456.

[138] R. D. Ball *et al.*, *Parton distributions for the LHC Run II*, JHEP **04**, 040 (2015), doi:10.1007/JHEP04(2015)040, arXiv:1410.8849.

[139] R. Brun, F. Bruyant, F. Carminati, S. Giani, M. Maire, A. McPherson, G. Patrick and L. Urban, *GEANT Detector Description and Simulation Tool* (1994), doi:10.17181/CERN.MUHF.DMJ1.

[140] T. Plehn, M. Spannowsky, M. Takeuchi and D. Zerwas, *Stop Reconstruction with Tagged Tops*, JHEP **10**, 078 (2010), doi:10.1007/JHEP10(2010)078, arXiv:1006.2833.

[141] T. Plehn and M. Takeuchi, *W+Jets at CDF: Evidence for Top Quarks*, J. Phys. G **38**, 095006 (2011), doi:10.1088/0954-3899/38/9/095006, arXiv:1104.4087.

[142] G. Cowan, *A survey of unfolding methods for particle physics*, Conf. Proc. C **0203181**, 248 (2002).

[143] F. Spano, *Unfolding in particle physics: a window on solving inverse problems*, EPJ Web Conf. **55**, 03002 (2013), doi:10.1051/epjconf/20135503002.

[144] M. Arratia *et al.*, *Publishing unbinned differential cross section results*, JINST **17**, P01024 (2022), doi:10.1088/1748-0221/17/01/P01024, arXiv:2109.13243.

[145] L. B. Lucy, *An iterative technique for the rectification of observed distributions*, The Astronomical Journal **79**, 745 (1974), doi:10.1086/111605.

[146] W. H. Richardson, *Bayesian-based iterative method of image restoration*, J. Opt. Soc. Am. **62**, 55 (1972), doi:10.1364/JOSA.62.000055.

[147] L. B. Lucy, *An iterative technique for the rectification of observed distributions*, Astron. J. **79**, 745 (1974), doi:10.1086/111605.

[148] G. D'Agostini, *A Multidimensional unfolding method based on Bayes' theorem*, Nucl. Instrum. Meth. A **362**, 487 (1995), doi:10.1016/0168-9002(95)00274-X.

[149] A. Hocker and V. Kartvelishvili, *SVD approach to data unfolding*, Nucl. Instrum. Meth. A **372**, 469 (1996), doi:10.1016/0168-9002(95)01478-0, arXiv:hep-ph/9509307.

[150] S. Schmitt, *TUnfold: an algorithm for correcting migration effects in high energy physics*, JINST **7**, T10003 (2012), doi:10.1088/1748-0221/7/10/T10003, arXiv:1205.6201.

[151] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn and R. Winterhalder, *How to GAN away Detector Effects*, SciPost Phys. **8**, 070 (2020), doi:10.21468/SciPostPhys.8.4.070, arXiv:1912.00477.

[152] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone and U. Köthe, *Invertible Networks or Partons to Detector and Back Again*, SciPost Phys. **9**, 074 (2020), doi:10.21468/SciPostPhys.9.5.074, arXiv:2006.06685.

[153] M. Backes, A. Butter, M. Dunford and B. Malaescu, *An unfolding method based on conditional invertible neural networks (cINN) using iterative training*, SciPost Phys. Core **7**, 007 (2024), doi:10.21468/scipostphyscore.7.1.007, arXiv:2212.08674.

[154] J. Ackerschott, R. K. Barman, D. Gonçalves, T. Heimel and T. Plehn, *Returning CP-observables to the frames they belong*, SciPost Phys. **17**, 001 (2024), doi:10.21468/SciPostPhys.17.1.001, arXiv:2308.00027.

[155] D. Maître and H. Truong, *A factorisation-aware Matrix element emulator*, JHEP **11**, 066 (2021), doi:10.1007/JHEP11(2021)066, arXiv:2107.06625.

[156] 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.

[157] 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.

[158] M. Backes, A. Butter, M. Dunford and B. Malaescu, *Event-by-event comparison between machine-learning- and transfer-matrix-based unfolding methods*, Eur. Phys. J. C **84**, 770 (2024), doi:10.1140/epjc/s10052-024-13136-3, arXiv:2310.17037.

[159] CMS Collaboration, *Measurement of the jet mass distribution and top quark mass in hadronic decays of boosted top quarks in proton-proton collisions at $\sqrt{s}$ = 13 TeV*, HEPData (collection), `https://doi.org/10.17182/hepdata.130712` (2023).

[160] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics* (2015), arXiv:1503.03585.

[161] G. Aad *et al.*, *Simultaneous Unbinned Differential Cross-Section Measurement of Twenty-Four Z+jets Kinematic Observables with the ATLAS Detector*, Phys. Rev. Lett. **133**, 261803 (2024), doi:10.1103/PhysRevLett.133.261803, arXiv:2405.20041.

[162] B. Nachman and V. Mikuni, *Large version of the omnifold dataset*, Zenodo, doi:10.5281/zenodo.10668638 (2024).

[163] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman and J. Thaler, *OmniFold: A Method to Simultaneously Unfold All Observables*, Phys. Rev. Lett. **124**, 182001 (2020), doi:10.1103/PhysRevLett.124.182001, arXiv:1911.09107.

[164] K. Desai, B. Nachman and J. Thaler, *Moment Unfolding* (2024), arXiv:2407.11284.

[165] A. J. Larkoski, S. Marzani, G. Soyez and J. Thaler, *Soft Drop*, JHEP **05**, 146 (2014), doi:10.1007/JHEP05(2014)146, arXiv:1402.2657.

[166] M. Dasgupta, A. Fregoso, S. Marzani and G. P. Salam, *Towards an understanding of jet substructure*, JHEP **09**, 029 (2013), doi:10.1007/JHEP09(2013)029, arXiv:1307.0007.

[167] J. Thaler and K. Van Tilburg, *Identifying Boosted Objects with N-subjettiness*, JHEP **03**, 015 (2011), doi:10.1007/JHEP03(2011)015, arXiv:1011.2268.

[168] R. T. Q. Chen, *torchdiffeq* (2018).

[169] A. Kendall and Y. Gal, *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* (2017), arXiv:1703.04977.

[170] S. Bollweg, M. Haußmann, G. Kasieczka, M. Luchmann, T. Plehn and J. Thompson, *Deep-Learning Jets with Uncertainties and More*, SciPost Phys. **8**, 006 (2020), doi:10.21468/SciPostPhys.8.1.006, arXiv:1904.10004.

[171] 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.

[172] 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.

[173] 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.

[174] 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.