SYMMETRY-AWARE NETWORKS FOR THE LHC

Dissertation

Jonas Simon Spinner

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

JONAS SIMON SPINNER

born in Wolfach, Germany

Oral examination: 09.07.2025

SYMMETRY-AWARE NETWORKS FOR THE LHC

Referees: Prof. Dr. Tilman Plehn
Prof. Dr. Ullrich Köthe

ABSTRACT

Of all scientific fields, few rival high-energy physics in the degree to which symmetries dictate its concepts, methods, and discoveries. Surprisingly, while particle-physics researchers were early enthusiasts of machine learning, they paid limited attention to models engineered around the discipline's built-in symmetries. Transformers implement the permutation symmetry of particles in an efficient and scalable way, yet their systematic application in high-energy physics is a recent development. We establish autoregressive transformers as event generators that grasp the autoregressive dynamics of QCD jet radiation and reliably generate jet multiplicities beyond the limits of the training data. Although various Lorentz-equivariant graph networks have been introduced for jet tagging, none employed Lorentz-equivariant transformer architectures. Our Lorentz-equivariant Geometric Algebra Transformer (L-GATr) closes this gap as the first Lorentz-equivariant transformer, matching the performance of the graph networks on small-scale datasets and outscaling them on large-scale datasets. Building on this foundation, we create the first Lorentz-equivariant generative network.

ZUSAMMENFASSUNG

In nur wenigen wissenschaftlichen Disziplinen spielen Symmetrien eine ähnlich dominante Rolle wie in Hochenergiephysik. Obwohl Teilchenphysiker schnell darin waren, Techniken des maschinellen Lernens auf ihr Feld anzuwenden, spielten Symmetrien zunächst eine untergeordnete Rolle. Transformer implementieren die Permutationssymmetrie zwischen Teilchen auf eine effiziente und skalierbare Art und Weise, dennoch ist deren systematische Anwendung in Hochenergiephysik ein noch junges Feld. Wir entwickeln autoregressive Transformer zur Generierung von LHC-Events, die die autoregressive Dynamik von QCD Strahlung implementieren und zuverlässig Multiplizitäten jenseits der Grenzen des Trainingsdatensatzes generieren können. Mehrere Lorentz-equivariante Graph-Netzwerke wurden für Jet-Klassifikation entwickelt, doch bisher gibt es noch kein Lorentz-equivariantes Netzwerk mit der Transformer-Architektur. Unser Lorentz-Equivarianter Geometrische Algebra Transformer (L-GATr) schließt diese Lücke als erster Lorentz-equivarianter Transformer, der auf kleinen Datensätzen ähnliche gute Ergebnisse wie die Graph-Netzwerke zeigt und sie auf großen Datensätzen übertrifft. Wir entwickeln das erste Lorentz-equivariante generative Netzwerk mithilfe der L-GATr-Architektur.

CONTENTS

PREFACE

The research presented in this thesis was conducted at the Institute for Theoretical Physics at Heidelberg University from October 2022 to December 2024. The contents of Chapters 4 to 6 are based on work in collaboration with other researchers and have been previously published as

[1] A. Butter et al., "Jet diffusion versus JetGPT – Modern networks for the LHC," SciPost Phys. Core **8**, 026 (2025), arXiv:2305.10475 [hep-ph].

[2] J. Spinner et al., "Lorentz-equivariant geometric algebra transformers for high-energy physics," Advances in Neural Information Processing Systems (2024), arXiv:2405.14806 [physics.data-an].

[3] J. Brehmer et al., "A Lorentz-Equivariant Transformer for All of the LHC," Submitted to SciPost Phys. (2024), arXiv:2411.00446 [hep-ph].

[4] A. Butter et al., "Extrapolating Jet Radiation with Autoregressive Transformers," Submitted to SciPost Phys. (2024), arXiv:2412.12074 [hep-ph].

Furthermore, the author contributed to the following publication. This work originated from research undertaken during his master's thesis at the Institute of Theoretical Particle Physics at the Karlsruhe Institute of Technology and was completed at the Institute for Theoretical Physics at Heidelberg University.

[1] C. A. Manzari et al., "Supernova limits on muonic dark forces," Phys. Rev. D **108**, 103020 (2023), arXiv:2307.03143 [hep-ph].

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

1 INTRODUCTION

Machine learning has become an indispensable tool in our effort to understand nature at its most fundamental level. Nowhere is this more evident than at CERN's Large Hadron Collider (LHC) – arguably the most complex machine ever built – where protons are accelerated to nearly the speed of light, collided, and their debris recorded by detector systems that generate on the order of 10^{15} bytes of raw data every second [1]. This torrent of information must be filtered, processed, and ultimately compared with theoretical predictions. Each stage involves decisions in extraordinarily high-dimensional spaces, decisions that today are often delegated to machine-learning algorithms, and increasingly to deep neural networks [2–4]. Such techniques have powered essentially every modern measurement in high-energy physics, culminating in the 2012 discovery of the Higgs boson [5, 6].

The demands placed on network architectures in this domain are severe. Models must natively encode the structure of particle-physics data while remaining expressive enough to capture intricate correlations in very high dimensions. Training samples often stem from computationally expensive first-principle calculations and detailed detector simulations, so data efficiency is paramount. Generative models face additional hurdles: detector acceptance boundaries and selection cuts introduce sharp edges in the probability density, yet accurate modeling of the extreme, low-density tails – sometimes spanning many orders of magnitude – is equally critical.

Although architectures first developed for vision or language tasks offer convenient starting points [7, 8], they seldom satisfy particle-physics requirements out of the box, largely because the inductive biases encoded in the underlying physics are not exploited. Particle interactions are governed by the symmetries of quantum field theory – most notably, permutation symmetry among particles and the Lorentz symmetry of special relativity [9]. In this thesis, networks are designed to be explicitly permutation- and Lorentz-equivariant and are applied to high-energy-physics problems. Particular emphasis is placed on uncertainty quantification with Bayesian neural networks (BNNs) to meet the field's stringent demands for control and precision.

This thesis is organized as follows. Chapter 2 introduces collider physics, outlines the LHC data-analysis workflow, and highlights the role of symmetries. Chapter 3 provides a concise overview of deep learning with special attention to permutation-equivariant architectures and, in particular, transformers – the principal architecture employed in this work. Chapter 4 combines a transformer with an autoregressive density-estimation scheme to generate reconstructed LHC events at high precision in a GPT-style

framework, with Bayesian uncertainty estimates incorporated. Chapter 5 extends this framework to learn event multiplicities; inspired by the autoregressive nature of QCD jet radiation, higher-multiplicity events are generated and extrapolations beyond the training range are explored. Chapter 6 introduces the first Lorentz-equivariant transformer as a versatile tool for collider physics and benchmarks it on regression, classification, and generative tasks, including the first Lorentz-equivariant generative network, which is compared with the earlier autoregressive models. Finally, Chapter 7 summarizes the results and outlines directions for future research.

2

COLLIDER PHYSICS

This chapter presents the fundamentals of LHC event generation and the underlying physics. An introduction to the Standard Model of high-energy physics is provided in Section 2.1. Representations of the primary collider-physics data type – particles – are examined in Section 2.2. The LHC analysis workflow and its machine-learning applications are described in Section 2.3. Finally, the importance of symmetries within this workflow is discussed in Section 2.4.

2.1 HIGH-ENERGY PHYSICS

Quantum field theory (QFT) is the foundation of modern high-energy physics. By specifying the relevant symmetry groups and the particles that inhabit them, one pins down all possible interaction types; the corresponding particle masses and coupling strengths then complete the model within the framework of quantum field theory. Currently, the Standard Model (SM) of particle physics is the most complete theory of high-energy physics. Its symmetry structure includes the Poincaré group, which underlies relativistic spacetime, and the gauge groups that describe the strong, weak, and electromagnetic interactions:

$$\mathrm{SU}(3)_c \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y. \tag{2.1}$$

Within this framework, the SM particles fall into three categories:

- GAUGE BOSONS A total of 12 gauge bosons mediate the interactions in Eq. (2.1): 8 gluons g for the strong force, the 3 heavy $W^{\pm}$ and Z bosons for the weak force, and the photon γ for electromagnetism.
- FERMIONS Three generations of fermions comprise 6 quarks (u, d, s, c, b, t), 3 heavy leptons (e, μ, τ), and their corresponding neutrinos (ν_e, ν_μ, ν_τ). All fermions except these neutrinos possess Dirac masses, while neutrinos stay massless in the SM.
- HIGGS BOSON The Higgs boson completes the Standard Model, providing a mechanism that explains the mass of all other particles.

Armed with these building blocks and 19 parameters, QFT can make first-principle predictions for all known fundamental interactions except gravity. The SM has been tested extensively—across collider experiments, cosmological observations, neutrino studies, and more. Yet, it still leaves some observed phenomena unexplained, most importantly:

- DARK MATTER Observations on galactic and cosmic scales demand a mysterious form of matter that only reveals itself through gravity.

- NEUTRINO MASS Studies on neutrinos from several sources provide evidence that neutrinos can not be exactly massless.
- MATTER-ANTIMATTER ASYMMETRY Our universe is dominated by matter, with no sizeable collections of antimatter. The SM offers no reason for this imbalance.

A huge variety of beyond-the-Standard-Model theories have been proposed to tackle these puzzles and some formal quirks of the SM related to a concept called naturalness. Modern high-energy physics research strives both to refine tests of the SM and to probe these extended frameworks. This thesis zeroes in on collider physics, examining how collider physics experiments can either confirm or challenge new ideas about our universe.

2.2 COLLIDER KINEMATICS

In the practical world of LHC physics, our usual language is 'particles in a detector'. A particle is characterized by its type and four-momentum

$$p = (E, \vec{p}) = (E, p_x, p_y, p_z). \tag{2.2}$$

The particle mass m appears via the Minkowski metric g

$$m^2 = E^2 - \vec{p}^2 = p^\mu p^\nu g_{\mu\nu} \quad \text{with} \quad g = \text{diag}(1, -1, -1, -1). \tag{2.3}$$

When working in a collider environment, it is natural to align the coordinate system with the collider topology. To this end, we define the z-axis as the beam direction. Further, it is natural to use the transverse momentum p_T and the azimuthal angle ϕ around the beam axis

$$p_T^2 = p_x^2 + p_y^2, \qquad \tan\phi = \frac{p_y}{p_x}. \tag{2.4}$$

To characterize the degree of alignment with the beam, we introduce the rapidity y, which adds linearly under Lorentz boosts, and its close relative, the pseudo-rapidity η

$$y = \frac{1}{2}\log\frac{E + p_z}{E - p_z}, \qquad \eta = \frac{1}{2}\log\frac{|\vec{p}| + p_z}{|\vec{p}| - p_z}. \tag{2.5}$$

Rapidity and pseudo-rapidity agree in the limit of small particle mass $m \ll E$ or $|\vec{p}| = E$ that is common in collider physics. The pseudo-rapidity can be measured to much higher precision because it does not depend on the particle energy E, making it a more practical choice.

Even though $\eta \in \mathbb{R}$, the pseudo-rapidity η and azimuthal angle ϕ are typically of similar magnitude in LHC physics. This motivates the definition of a simple measure for the angular seperation of two particles

$$\Delta R_{ij}^2 = (\phi_i - \phi_j)^2 + (\eta_i - \eta_j)^2. \tag{2.6}$$

For practical purposes, the parametrization (p_T, ϕ, η, m) is more useful than four-momenta (E, p_x, p_y, p_z). Hence, the former will serve as the primary momentum representation throughout this thesis.

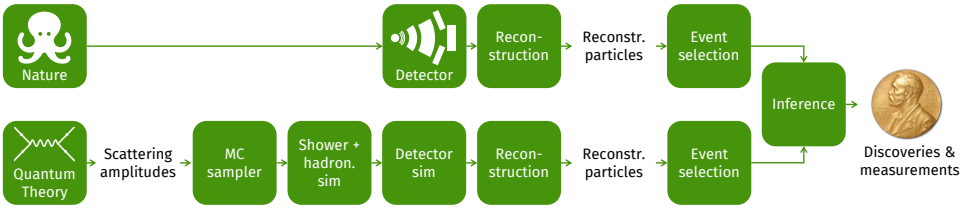

Figure 2.1: Schematic view of the data-analysis workflow in high-energy physics. Measurements (top) are processed in parallel with simulated data (bottom). Their comparison is ultimately the basis for most scientific conclusions.

2.3 THE LHC WORKFLOW

The Large Hadron Collider (LHC), located in Geneva, Switzerland, is the largest operational particle collider. It hosts four main experiments: the multi-purpose ATLAS and CMS detectors, the LHCb experiment focusing on flavor physics, and the ALICE experiment specializing in heavy-ion physics. With collisions occuring every 30 ns, these experiments collect roughly 10^{15} bytes of data every second [1]. By the end of its lifetime, the LHC is expected to produce a dataset on the order of 10^9 GB. Analyzing such a vast amount of information presents a remarkable opportunity for high-energy physics – and calls for especially robust and efficient analytical tools.

In fact, the LHC is the largest machine built by humankind.

2.3.1 *LHC data analysis workflow*

The data-analysis workflow in LHC physics is illustrated in Figure 2.1. The central idea is to take both the collected detector data and the predictions from various theoretical models, process them in parallel, and ultimately compare the resulting distributions. The main steps are outlined below, moving from left to right in Figure 2.1.

This is the most basic workflow. There are more approaches, e.g. anomaly detection, unfolding and data-driven analysis techniques.

MATRIX ELEMENT Starting with the Lagrangian and its associated parameters, one uses QFT to compute the matrix element for a given process and up to a given order in perturbation theory. Modern event generators like MadGraph [10] and Sherpa [11] automate this procedure for tree-level and one-loop processes. A key challenge is that the number of diagrammatic contributions increases factorially with the number of final-state particles, making this stage time-consuming for complex processes and higher orders in perturbation theory.

MONTE CARLO SAMPLING Next, one samples LHC events from the probability distribution defined by the matrix element and phase-space factors. Events serve as a more flexible representation than the corresponding probabilities, a requirement for the next simulation steps. Sampling from a high-dimensional distribution is a common task in statistics, and Monte Carlo sampling is the standard tool to tackle it. In LHC physics, the proposal distribution is constructed based on process-specific phase space

mappings and a base distribution which is dynamically adapted using the VEGAS algorithm [12]. Small discrepancies in the proposal distribution require an additional unweighting step.

In higher-order calculations, one has to be careful to not double-count emissions in matrix element and parton shower.

PARTON SHOWER High-energetic quarks and gluon radiation appears in the matrix element at higher orders in perturbation theory, while soft and collinear emissions are handled by parton shower algorithms. In the soft or collinear limit, the $n+1$-particle cross-section factorizes into the n-particle cross-section and a universal splitting kernel [13]. This property underpins all parton shower approaches, where successive splittings continue until a cutoff scale is reached—beyond which QCD is non-perturbative.

HADRONIZATION Once the momentum scale drops below $\Lambda_{\text{QCD}} \sim 100$ GeV, partons become strongly coupled and cluster into color-neutral hadrons. First-principle methods cannot describe this transition, so phenomenological models are employed instead, with the Lund string model [14, 15] and the cluster model [16] among the most widely used. Finally, the decay of unstable hadrons is simulated. These stages all occur on timescales too short to be resolved in a detector, whereas subsequent processes unfold at macroscopic scales.

DETECTOR SIMULATION Within particle detectors, traversing particles are converted into measurable electronic signals. General-purpose detectors such as CMS [17] and ATLAS [18] typically combine a tracking system to measure the momentum of charged particles, electromagnetic and hadronic calorimeters for energy measurements, and a muon system. Information from each subsystem is combined to reconstruct the final-state particles [19, 20]. Detector simulations that accurately capture these effects are computationally expensive and often dominate the computing budget in LHC analyses.

JET RECONSTRUCTION In many analyses, the large number of hadrons produced via QCD processes is not of primary interest. Instead, hadrons are clustered into jets – groups of particles that approximate the momentum of an underlying hard-scatter particle. Various jet algorithms exist; a widely used option is the anti-k_T algorithm [21]. It sequentially merges particles based on the distance between particles d_{ij} and the distance to the beam d_{iB}

$$d_{ij} = \min(p_{T,i}^{-2}, p_{T,j}^{-2}) \frac{\Delta R_{ij}^2}{R^2} \quad \text{and} \quad d_{iB} = p_{T,i}^{-2}, \tag{2.7}$$

where R (commonly $0.4 - 1.0$) controls the jet cone size, and ΔR_{ij} is the angular distance between particles i and j. At each iteration, the minimum of all d values is found; if it corresponds to d_{ij}, the particles i and j are merged, or if it corresponds to $d_{i,B}$, particles i is identified as a jet. Intuitively, the algorithm combines high-p_T particles with their lower-p_T neighbours until only well-seperated jets of significant transverse momentum remain.

EVENT RECONSTRUCTION In the final step, the reconstructed detector objects can be matched to particles in the original (hard) process. This task is inherently ambiguous, since the trajectory from partons to detector signals

is only partially observable. Event reconstruction often proceeds by fitting decay chains to invariant-mass constraints, while neutrinos are inferred from the missing transverse energy. Depending on the analysis objectives, event reconstruction may be optional or only partially feasible.

EVENT SELECTION Finally, the LHC dataset is vast, and using the entire dataset is neither practical nor necessary for most analyses. Instead, the data is subjected to multiple selection stages to filter out only the most relevant events. An online trigger system decides which collisions are written to disk, and each analysis further refines its selection criteria to maximize the relevant information in the final dataset.

INFERENCE After reconstruction and event selection are applied to both measured data and simulated events, the framework of hypothesis testing provides a systematic way to assess whether the two sets agree or not. Any significant discrepancy can hint at unaccounted-for effects in the simulation, or physics beyond the Standard Model.

2.3.2 *Machine Learning enhancements*

Recent advances in machine learning, combined with the rapidly growing volume of collider data, demand improvements to the workflow outlined above. A logical first step is to optimize existing methods – for example, by parallelizing operations and taking advantage of graphical processing units (GPUs). Beyond that, machine learning (ML) offers a powerful suite of big-data techniques that can further enhance the efficiency of the workflow. Below, we highlight some prominent ML applications in LHC analyses, loosely divided into cases where ML is used to improve the modelling or speed up the simulation.

MODELLING Many operations in the LHC workflow suffer from either suboptimal numerical techniques or an incomplete understanding of the underlying physics, like non-perturbative effects. While traditional methods can address these issues to some extent, modern machine learning approaches have the potential to significantly advance the state of the art. Below is a selection of ML applications that illustrate how these techniques can lead to more efficient or more accurate modeling:

- NEURAL IMPORTANCE SAMPLING One key factor in the efficiency of importance sampling is the quality of the proposal distribution. While classical techniques such as the VEGAS algorithm remain widely used, generative neural networks can produce more accurate proposal distributions – albeit at the expense of a longer initial training phase. In particular, normalizing flow–based methods have demonstrated substantial speed-ups for complex processes, highlighting their potential to significantly reduce computational overhead [22–24].

- HADRONIZATION MODELS Machine learning offers a versatile, highly tunable framework for developing empirical models of hadronization. [25, 26].

- RECONSTRUCTION ALGORITHMS Machine learning can enhance reconstruction algorithms at every stage of the data-processing chain – from track reconstruction and the identification of individual particles to full event reconstruction [27].

- JET TAGGING Classifying particle jets according to their originating particle is a routine procedure in nearly every LHC analysis. Today, it stands as one of the most mature and widely adopted ML applications in collider physics [28, 29].

- SIMULATION-BASED INFERENCE Simulation-based inference is a modern parameter-estimation approach in which neural networks – classifiers or generators – relate the likelihood and posterior distribution. Given the well-developed simulation frameworks in LHC physics, this methodology finds a particularly natural application in collider analyses [30].

- UNFOLDING Unfolding aims to reconstruct underlying events from the measured detector signals – even reverting back to a stage before the parton shower if necessary. This can be accomplished via classifier-based reweighting or generative models, trained on simulated events and subsequently applied to real data [31–33].

- MATRIX ELEMENT METHOD The matrix element method is widely regarded as the optimal analysis technique in LHC physics. It involves computing a phase space integral for every observed event, a process that rapidly becomes unfeasible as the event count grows. Machine learning techniques – especially the combination of neural importance sampling with conditional generative networks – hold promise for making this method both sufficiently expressive and computationally practical [34, 35].

- ANOMALY DETECTION One of the central missions of the LHC is to hunt for anomalous high-energy phenomena – a difficult task, given that "anomalous" is not strictly defined. Machine learning circumvents this ambiguity by assigning anomaly scores, offering a model-agnostic route to uncovering potential new physics [36].

SPEED Another branch of ML applications aims to accelerate parts of the simulation chain that are rooted in first principles but are computationally intensive. Neural networks that bypassing the first-principle simulations can significantly reduce computation times, provided they are sufficiently expressive so that their modeling uncertainty remains negligible. These neural surrogates have to be trained on a dataset that is sufficiently large for the required precision of the task [37, 38].

- AMPLITUDE REGRESSION Matrix element, or amplitude, regression, refers to the process of fitting a neural network surrogate to the analytic expression of the matrix element. This technique becomes particularly valuable at higher orders in perturbation theory, where direct evaluation of the matrix element is computationally demanding [39, 40].
- DETECTOR SIMULATION Detector simulation constitutes one of the largest contributions to the overall computational cost in LHC simulations. Moreover, it represents the most established application of generative networks in LHC physics [41].
- EVENT GENERATION Generative networks can be employed to bypass the entire simulation chain, enabling a direct mapping from the process specification to reconstructed events [8, 42, 43].

This thesis focuses on machine learning applications in the second class. In the Chapters 4 and 5 we discuss how autoregressive transformers can be used to construct event generators. In Chapter 6 we introduce a Lorentz-equivariant transformer and apply it to amplitude regression, jet tagging and event generation.

2.4 SYMMETRIES IN THE LHC WORKFLOW

Being based on quantum field theory, LHC physics is inherently governed by a host of symmetries, many of which manifest throughout the analysis workflow. To build both expressive and efficient tools, it is crucial to embed these symmetry principles directly into our methods. Demonstrating how to achieve this constitutes the central objective of this thesis.

In LHC data, the fundamental object is the particle—defined by its discrete type and its four-momentum. Particles of the same type are indistinguishable, leading to an exact permutation symmetry. Permutation-equivariant architectures – well established in the machine-learning community – are therefore a natural fit for particle-based analyses. In Section 3.2, we will review these architectures with a focus on transformers.

Meanwhile, each particle's four-momentum transforms under the Lorentz group's boosts and rotations. In the remainder of this section, we provide a concise overview of the Lorentz group's properties and discuss cases of Lorentz symmetry breaking in the LHC workflow.

LORENTZ SYMMETRY The laws of fundamental physics are invariant with respect to the choice of an inertial reference frame: they do not change under rotations and boosts from one un-accelerated reference frame into another [44]. Together, these transformations form the special orthochronous Lorentz group $\mathrm{SO}^+(1,3)$."Special" and "orthochronous" here mean that spatial and temporal reflections are not considered as symmetries. In fact, the fundamental laws of nature are *not* invariant under those transformations, an effect known as P-violation [45] and T-violation [46].

Allowing for accelerating reference frames would bring us to the general theory of relativity.

The special orthochronous Lorentz group is the connected component of the orthogonal group on the four-vector space $\mathbb{R}^{1,3}$ with Minkowski metric $\mathrm{diag}(+1,-1,-1,-1)$. Lorentz transformations mix temporal and spatial components. Space and time should therefore not be considered as separate concepts, but rather as components of a four-dimensional space-time. Particle four-momenta are another instance of this: they transform in the vector representation of the Lorentz group, with the Lorentz transformation mixing energy and spatial momentum

$$p^\mu \to p'^\mu = \Lambda^\mu_\nu p^\nu, \qquad \Lambda \in SO^+(1,3). \tag{2.8}$$

LORENTZ SYMMETRY BREAKING While the underlying laws of physics are Lorentz invariant, various aspects of the LHC measurement process explicitly break this symmetry. Key examples include:

- PROTON BEAM The beam axis is a fixed direction in the laboratory frame. One may formally restore invariance by treating the incoming protons as additional particles, but in practice the beam direction singles out a preferred axis.
- DETECTOR GEOMETRY The complex arrangement of sensors—layers of trackers, calorimeters, and muon chambers—is not invariant under arbitrary rotations or Lorentz boosts. For example, particles striking the central (barrel) region are typically measured more precisely than those entering at shallow angles to the beam. However, an approximate rotational symmetry around the beam axis remains.
- JET ALGORITHM Jet reconstruction algorithms define jets as collimated boosted objects. For instance, the anti-k_T algorithm [21] uses the transverse momentum p_T and the angular seperate ΔR to define the distance between particles (2.7). Since p_T and ΔR are not invariant under longitudinal boosts, the resulting jet assignments change with the reference frame.

Understanding which symmetries are broken (and which persist) is essential when designing Lorentz-aware network architectures. Imposing too large symmetry groups can render the model incapable of distinguishing physically distinct configurations, thereby degrading performance.

3 MACHINE LEARNING

The foundational knowledge required for the machine-learning applications treated in the later chapters is presented in this chapter. A concise introduction to deep learning is given in Section 3.1. Lastly, permutation-equivariant architectures are examined in Section 3.2, with particular emphasis placed on transformers, the principal architecture employed in this thesis.

3.1 DEEP LEARNING

It is called deep *learning, because neural networks can be stacked to large depth.*

The strategy in modern machine learning is to embed data into a high-dimensional latent space, process it with a neural network, and extract the network output for use in downstream applications. The network is trained to minimize a loss function, which contains information about the desired network properties. Training is performed with stochastic gradient descent, a simple but efficient algorithm that scales well to large networks.

MULTILAYER PERCEPTRON The bread and butter of deep learning are fully-connected networks, or multi-layer perceptions (MLPs). A single perceptron can be written as

$$x'_c = \phi\left(W_{cc'}x_{c'} + b_c\right), \tag{3.1}$$

where $W \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$ are free parameters and $x, x' \in \mathbb{R}^n$ is the data embedded in a high-dimensional latent space. The nonlinearity ϕ is required to obtain a universal approximator [47]. A multilayer perceptron is a sequence of tens or hundreds of such perceptron operations that modify the latent representation x. In practice, the minimal setup (3.1) is modified with concepts like residual connections, dropout and normalization techniques to improve the performance. Moreover, one can construct architectures that mimic the properties of the data, as in convolutional neural networks (CNNs) for grids, and permutation-equivariant archiectures for point clouds.

LOSS FUNCTION But what is a good choice for the free parameters W, b in (3.1)? Consider for instance a regresion task, where the dataset consists of pairs $(x_i, y_i), i = 1 \dots N$ and the target is to fit a neural network f such that $f(x_i) = y_i$. A common objective or loss function for this task is the mean squared error (MSE)

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2 = \left\langle (y - f(x))^2 \right\rangle_{x,y}. \tag{3.2}$$

Note that the losses $\alpha\mathcal{L}$ and $\mathcal{L} + \alpha$ with $\alpha \in \mathbb{R}$ are equivalent to $\mathcal{L}$.

A more principled approach to loss functions is to tackle the problem from a probabilistic perspective. The data is described by a distribution $p_{\text{data}}(x)$, and the network output is described by $p_{\text{model}}(x)$. The training objective is to match these distributions, i.e. $p_{\text{data}}(x) = p_{\text{model}}(x)$. The Kullback-Leibler (KL) divergence is a distance measure between two distributions, serving as a flexible blueprint for loss functions

$$\begin{aligned} \mathrm{KL}(p_{\text{data}}|p_{\text{model}}) &= \int dx p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{model}}(x)} \\ &= -\int dx p_{\text{data}}(x) \log p_{\text{model}}(x) + \text{const.} \end{aligned} \tag{3.3}$$

For instance, the MSE loss (3.2) can be derived from a gaussian distribution $p_{\text{model}}(x|y) = \mathcal{N}_{y,1}(x)$, and the cross-entropy (CE) loss commonly used for classification tasks can be derived from a multinomial distribution. Generative networks do not constrain the form of $p_{\text{model}}(x)$, and hence they can be directly trained on the KL divergence (3.3).

All projects in this thesis use the PyTorch *[48] autodiff library.*

GRADIENT DESCENT So how can we find the neural network parameters that minimize the loss function? Gradient descent is a simple optimization algorithm that scales well to high-dimensional systems. It relies on automatic differentiation (autodiff), a numerical method to efficiently evaluate gradients $\nabla_\theta \mathcal{L}$ of the objective $\mathcal{L}$ with respect to the free parameters of the problem θ. These gradients are then used to update the parameters in the direction of the gradient descent $-\nabla_\theta \mathcal{L}$,

$$\theta' = \theta - \lambda \nabla_\theta \mathcal{L}. \tag{3.4}$$

A popular learning rate is $\lambda = 0.0003$ *[49].*

The learning rate λ is a hyperparameter that quantifies the greediness of the optimization algorithm. Typically, the gradient $\nabla_\theta \mathcal{L}$ is only evaluated over a random subset of the dataset or batch to inrease randomness, this is called stochastic gradient descent. There are refined version of the basic gradient descent (3.4) that incorporate also gradient information of previous update steps to avoid convergence to local minima of the objective. Examples are the so-called Adam [50], RAdam [51] and Lion [52] optimizers.

BAYESIAN NEURAL NETWORKS The training process of a neural network introduces uncertainties, and the resulting predictions have little value if these uncertainties are not quantified. A first approach is ensembling, i.e. training multiple models and estimating the models uncertainty based on the distribution of predictions. A more principled and scalabled approach are Bayesian neural networks (BNNs) [53–56]. We assign a prior distribution of the network parameters $p(\theta)$, and ask for the posterior distribution $p(\theta|\mathcal{D})$ of network parameters given the training data $\mathcal{D}$. The key idea is to approximate the intractable posterior distribution $p(\theta|\mathcal{D})$ with a simpler distribution $q_\phi(\theta)$ that depends on free parameters ϕ. The parameters ϕ can be optimized with a KL divergence as the objective

The BNN prior is handled like a hyperparameter.

$$\mathcal{L}_{\text{BNN}} = \mathrm{KL}\left(q_\phi(\theta)|p(\theta|\mathcal{D})\right). \tag{3.5}$$

We now use Bayes theorem to rewrite the BNN objective in terms of the model likelihood $p_{\text{model}}(\mathcal{D}|x)$

BNNs estimate uncertainties at the cost of extra free parameters ϕ.

$$\begin{aligned}\mathcal{L}_{\text{BNN}} &= \int d\theta q_\phi(\theta) \log \frac{q_\phi(\theta)}{p(\theta|\mathcal{D})} = \int d\theta q_\phi(\theta) \log \frac{q_\phi(\theta) p(\mathcal{D})}{p(\mathcal{D}|\theta) p(\theta)} \\ &= -\int d\theta q_\phi(\theta) \log p(\mathcal{D}|\theta) + \text{KL}\left(q_\phi(\theta)|p(\theta)\right) + \log p(\mathcal{D}). \end{aligned} \tag{3.6}$$

In the resulting expression the first term is a modification of the usual objective $-\log p(\mathcal{D}|\theta)$, the second term acts as a regularization for the model parameters θ, and the third term is a constant that does not affect the optimization. The integral in the first term is typically approximated with a single sample in each iteration. In conclusion, training BNNs amounts to sampling model parameters $\theta \sim q_\phi(\theta)$ from the approximate posterior in each iteration, and adding an extra loss term that regulates the posterior distribution $q_\phi(\theta)$.

More expressive parametrizations $q_\phi(\theta)$ are usually more expensive to optimize.

But how what is a good parametrization for the approximate likelihood $q_\phi(\theta)$? A common choice are uncorrelated normal distributions $\mathcal{N}_{\mu_i,\sigma_i}$ for the network parameters θ_i. This seemingly restrictive ansatz can still model complex uncertainties if the network is sufficiently deep. The prior is typically chosen to be a normal distribution with zero mean and a tuneable standard deviation σ_p. The KL divergence in the BNN objective (3.6) can then be evaluated analytically

$$\text{KL}\left(q_\phi(\theta)|p(\theta)\right) = \sum_i \left(\frac{\mu_i^2 + \sigma_i^2 - \sigma_p^2}{2\sigma_p^2} + \log \frac{\sigma_p}{\sigma_i} \right). \tag{3.7}$$

3.2 PERMUTATION-EQUIVARIANT ARCHITECTURES

The MLPs introduced above are sufficient to construct universal approximators in the limit of infinite width [47], but in practice finite-width MLPs do not give sufficient performance for most tasks. In most applications there are inherent symmetries in the data, and reflecting these symmetries in the choice of the network architecture can significantly boost performance. For instance, the discrete translation symmetry of grids is naturally represented in convolutional neural networks, which are the typical architecture for image processing tasks. We will focus on permutation symmetry in this section, and discuss the case of Lorentz symmetry in Chapter 6.

PERMUTATION EQUIVARIANCE The underlying principle is called *equivariance*. We call an operation f *equivariant* under the symmetry group $\mathcal{G}$, if for any possible input x and any transformation $t \in \mathcal{G}$

Equivariant networks transform as tensors of the respective symmetry group.

$$f(t(x)) = t(f(x)). \tag{3.8}$$

For the case of permutation symmetry, t is a permutation and x is a set. *Invariant* operations f are a subclass of equivariant operations where the output $f(x)$ does not change under transformations $t \in \mathcal{G}$,

$$f(t(x)) = t(f(x)) = f(x). \tag{3.9}$$

Equivariance is typically achieved by constraining the available operations. For instance, permutation-equivariant architectures usually apply the same learnable operations to each set element individually, and then exchange information between set elements using simple permutation-invariant operations like summing or taking the maximum over set elements. A very general class of permutation-equivariant operations are message-passing graph networks which are based on the operation [57]

$$x_i' = \phi\left(x_i, \bigoplus_j \psi(x_i, x_j)\right), \tag{3.10}$$

where x_i are embeddings of the set elements, ϕ, ψ are unconstrained neural networks, and $\bigoplus$ is a permutation-invariant aggregation operation like $\sum$ or max. In words, these networks create updates of hidden states x_i' based on the previous states x_i and messages $\psi(x_i, x_j)$ between pairs of set elements which are aggregated in a permutation-invariant way.

Enforcing equivariance to symmetries present in the data typically improves performance for two reasons. First, the neural network does not have to learn the symmetry properties of the output, saving parameters for the actual task. Second, equivariant architectures typically have parameter sharing, i.e. the same free parameters are used in different parts of the network. This acts as an efficient regulator to avoid overfitting.

TRANSFORMERS Transformers [58] are one kind of permutation-equivariant architecture that empirically shows excellent scaling properties when large sets or large latent representations are required. They are based on *self-attention* [58], which emerges as a special case of (3.10)

$$x_{ic}' = \sum_j A_{ij} v_{jc} = \sum_j \mathrm{Softmax}_j\left(\sum_{c_1=1}^{d} \frac{q_{ic_1} k_{jc_1}}{\sqrt{d}}\right) v_{jc}, \tag{3.11}$$
$$\text{with} \quad q_{ic} = W^Q_{cc'} x_{ic'}, \quad k_{ic} = W^K_{cc'} x_{ic'}, \quad v_{ic} = W^V_{cc'} x_{ic'}.$$

The matrices $W^Q, W^K, W^V \in \mathbb{R}^{d\times d}$ are learnable. The softmax operation is used to normalized the attention matrix A_{ij}, and the scaling factor $\sqrt{d}$ cancels the scaling of the inner product $q_{ic_1} k_{jc_1}$ with the latent dimension d. In practice, h distinct sets of d/h-dimensional keys, queries and values are constructed from the d-dimensional latent representation x and again aggregated after attention to increase expressivity, this is called *multi-head* attention.

Intuitively, attention compares queries q with keys k to construct the attention matrix A_{ij} that quantifies the interaction between any pair of set elements. This matrix is then matched onto values v to obtain the updated set elements x'. Attention can also be used to include information from elements of another set y_{ic} into the updates x_{ic}' by setting $k_{ic} = W^K_{cc'} y_{ic'}, v_{ic} = W^V_{cc'} y_{ic'}$ instead, this is called cross-attention [58].

The superior scaling properties of attention (4.1) is due to the fact that attention only involves learnable operations on node representations x_i, and

not on edge representations $x_i x_j$ which are included as $\psi(x_i, x_j)$ in the more general message-passing operation (3.10). This allows the construction of fused attention kernels [59, 60], which significantly improve the speed and memory consumption of attention.

Transformers are broadly defined as permutation-equivariant architectures that use attention as the only message-passing operation [58]. They typically feature a series of blocks which combine attention with a two-layer MLP to post-process the attention outputs. Layer normalization [61] and residual connections [62] are typically added to improve training stability when scaling to many layers.

Language is not permutation equivariant: "this is correct" ≠ "is this correct".

PERMUTATION SYMMETRY BREAKING Transformers were developed for language processing, a field where the relevance of permutation equivariance is not immediately obvious. They were proposed in combination with positional encodings [58], i.e. extra input features that are different for each word. This is a form of *permutation equivariance breaking*, which is necessary in this case to reflect the properties of the data.

Even image patches can be interpreted as a set using permutation symmetry breaking [63].

In high-energy physics, an exact permutation symmetry is only present between identical particles. In practice it often is beneficial to instead consider the extended group of permutations of all particles, including those of different kind, and then break this symmetry in a way similar to language modeling. Usually only a fixed number of particles is present, allowing a one-hot encoding as a simple way to break this larger symmetry group down to the exact subgroup.

We emphasize that the examples of symmetry breaking discussed above do not affect the exact permutation equivariance in the architecture. Instead, they break permutation equivariance at the level of the inputs by adding extra reference information onto each set element. In principle this design allows the network to restore permutation equivariance if the data support it, by mapping two different reference inputs to the same point in latent space. This recipe for symmetry breaking in equivariant architectures can be applied generally. We will see more examples in Section 6.1.3 for the example of Lorentz-equivariant architectures.

4

LHC EVENT GENERATION WITH JETGPT

The research presented in this chapter is based on work in collaboration with Anja Butter, Nathan Huetsch, Sofia Palacios Schweitzer, Tilman Plehn and Peter Sorrenson, and has been previously published in Ref. [8]. All tables and figures as well as parts of the text are similar or identical to the content of these articles.

At the LHC, generative networks are used for many simulation and analysis tasks, typically to describe virtual or real particles over their correlated phase space. The number of particles ranges from few to around 50, described by their energy and three momentum directions, sometimes simplified through on-shell conditions.

Several generative networks have been applied to this task. First, generative adversarial networks (GANs) have been used to show that the distribution of LHC events can be tackled with neural networks [42], although not to the necessary level of precision. Bijective networks like INNs are a better alternative, being able to learn most distribution to percent-level accuracy [43]. In this chapter, we develop an autoregressive transformer as an alternative that leverages the approximate permutation symmetry between particles to build more meaningful internal representations. It was developed jointly with a denoising diffusion probabilistic model and a conditional flow matching model, see reference [64] for the comparison.

Because fundamental physics applications require full control and a conservative and reliable uncertainty estimation of neural networks, we will develop a Bayesian version the generative transformer. This allows us to control the uncertainty in the density estimation and to derive an intuition how the different networks learn the phase space distribution of the data.

4.1 GENERATIVE MODEL

Architecture

A distinct shortcoming of traditional generative models like GANs, INNs, and diffusion models is that they learn the correlations in all phase space directions simultaneously. This leads to a power-law scaling for instance of the training effort for a constant precision in the learned correlations [39]. The autoregressive transformer (AT) [65] instead interprets the phase space vector $x = (x_1, ...x_n)$ as a sequence of elements x_i and factorizes the joint n-dimensional probability into n probabilities with a subset of conditions,

$$p_{\text{model}}(x|\theta) = \prod_{i=1}^{n} p(x_i|x_1, ..., x_{i-1}) \approx p_{\text{data}}(x) \,, \tag{4.1}$$

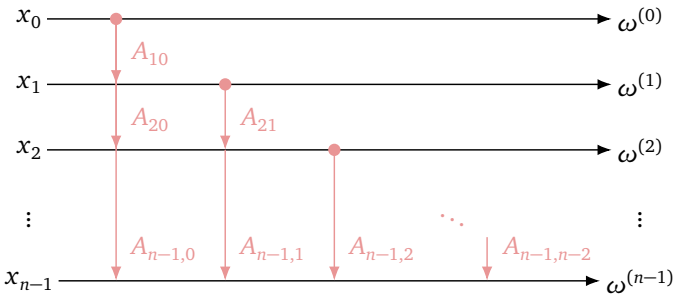

Figure 4.1: Autoregressive approach to density estimation. The attention matrix A_{ij} defined in Eq.(4.8) encodes information between components x_i. We introduce an auxiliary condition $x_0 = 0$ for the first phase space component x_1.

as illustrated in Fig. 4.1. This autoregressive approach improves the scaling with the phase space dimensionality in two ways. First, each distribution $p(x_i|x_1,...x_{i-1})$ is easier to learn than a distribution conditional on the full phase space vector x. Second, we can use our physics knowledge to group challenging phase space directions early in the sequence $x_1,...,x_n$.

The network learns the conditional probabilities over phase space using a representation

$$p(x_i|\omega^{(i-1)}) = p(x_i|x_1,...x_{i-1}) \,, \tag{4.2}$$

where the parameters $\omega^{(i-1)}$ encode the conditional dependence on $x_1,...x_{i-1}$. A naive choice are binned probabilities $w_j^{(i-1)}$ per phase space direction,

$$p(x_i|\omega^{(i-1)}) = \sum_{\text{bins } j} w_j^{(i-1)} \mathbb{1}^{(j)}(x_i) \,, \tag{4.3}$$

where $\mathbb{1}^{(j)}(x)$ is one for x inside the bin j and zero outside. A more flexible and better-scaling approach is a Gaussian mixture,

$$p(x_i|\omega^{(i-1)}) = \sum_{\text{Gaussian } j} w_j^{(i-1)} \mathcal{N}(x_i; \mu_j^{(i-1)}, \sigma_j^{(i-1)}) \,. \tag{4.4}$$

It generalizes the fixed bins to a set of learnable means and widths.

Our architecture closely follows the Generative Pretrained Transformer (GPT) models [65], illustrated in Fig. 4.2. The network takes a sequence of x_i as input and evaluates them all in parallel. We use a linear layer to

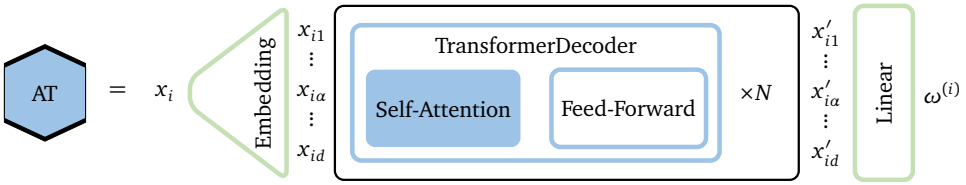

Figure 4.2: Architecture of the autoregressive transformer. All phase space components x_i are evaluated in parallel, see Fig. 4.1.

map each value x_i in a d-dimensional latent space, denoted as $x_{i\alpha}$. The network consists of a series of TransformerDecoder blocks, combining a self-attention layer with a standard feed-forward network. Finally, a linear layer maps the latent space onto the representation $\omega^{(i-1)}$ of the conditions.

Equations (4.3) and (4.4) do not provide an actual structure correlating phase space regions and phase space directions. This means the transformer needs to construct an appropriate basis and correlation pattern by transforming the input x into an x', with the same dimension as the input vector and leading to the ω representation. Its goal is to construct a matrix A_{ij} that quantifies the relation or similarity of two embedded phase space components $x_{i\alpha}$ and $x_{j\alpha}$. We construct the single-headed self-attention [58] of an input x in three steps.

1. Using the conventions of the first layer, we want to measure the relation between x_i and a given x_j, embedded in the d-dimensional latent space. Replacing the naive scalar product $x_{i\alpha}x_{j\alpha}$, we introduce learnable latent-space transformations $W^{Q,K}$ to the elements

$$q_{i\alpha} = W^Q_{\alpha\beta} x_{i\beta} \qquad \text{and} \qquad k_{j\alpha} = W^K_{\alpha\beta} x_{j\beta} \;, \tag{4.5}$$

   and use the directed scalar product

$$A_{ij} \sim q_{i\alpha} k_{j\alpha} \tag{4.6}$$

   to encode the relation of x_j with x_i through k_j and q_i. While the scalar product is symmetric, the attention matrix does not have to be, $A_{ij} \neq A_{ji}$. These global transformations allow the transformer to choose a useful basis for the scalar product in latent space.

2. The first problem with A_{ij} given in Eq.(4.6) is that it grows with the latent space dimensionality, so it turns out to be useful to replace it by $A_{ij} \to A_{ij}/\sqrt{d}$. More importantly, we want all entries j referring to a given i to be normalized,

$$A_{ij} \in [0,1] \qquad \text{and} \qquad \sum_j A_{ij} = 1 \;. \tag{4.7}$$

   This leads us to the definition

$$A_{ij} = \text{Softmax}_j \frac{q_{i\alpha} k_{j\alpha}}{\sqrt{d}} \quad \text{with} \quad \text{Softmax}_j(x_j) = \frac{e^{x_j}}{\sum_k e^{x_k}} \;. \tag{4.8}$$

   Similar to the adjacency matrix of a graph, this attention matrix quantifies how closely two phase space components are related. Our autoregressive setup sketched in Fig. 4.1 requires us to set

$$A_{ij} = 0 \qquad \text{for} \quad j > i \;. \tag{4.9}$$

3. Now that the network has constructed a basis to evaluate the relation between two input elements x_i and x_j, we use it to update the actual

representation of the input information. We combine the attention matrix A_{ij} with the input data, but again transformed in latent space through a learnable matrix W^V,

$$\begin{aligned} x'_{i\alpha} &= A_{ij} v_{j\alpha} = A_{ij} W^V_{\alpha\beta} x_{j\beta} \\ &= \text{Softmax}_j \left(\frac{W^Q_{\delta\gamma} x_{i\gamma} W^K_{\delta\sigma} x_{j\sigma}}{\sqrt{d}} \right) W^V_{\alpha\beta} x_{j\beta} \; . \end{aligned} \tag{4.10}$$

In this form we see that the self-attention vector x' just follows from a general basis transformation with the usual scalar product, but with an additional learned transformation for every input vector.

The self-attention can be stacked with other structures like a feed-forward network, to iteratively construct an optimal latent space representation. This can either be identified with the final output $\omega^{(i)}$ or linked to this output through a simple linear layer. To guarantee a stable training of this complex structure, we evaluate the self-attention as an ensemble, defining a multi-headed self-attention. In addition, we include residual connections, layer normalization, and dropout just like the GPT model. Because the sum over j in Eq.(4.10) leads to permutation equivariance in the phase space components, we break it by providing explicit positional information through a linear layer that takes the one-hot encoded phase space position i as input. This positional embedding is then added to the latent representation $x_{i\alpha}$.

Training and sampling

The training of the autoregressive transformer is illustrated in Fig. 4.3. We start with an universal $x_0 = 0$ in $p(x_1|\omega^{(0)})$ for all events. The transformer encodes all parameters ω needed for $p(x_i|\omega^{(i-1)})$ in parallel. The chain of conditional likelihoods for the realized values x_i gives the full likelihood $p_{\text{model}}(x|\theta)$, which in turn can be used for the loss function

$$\begin{aligned} \mathcal{L}_{\text{AT}} &= \Big\langle -\log p_{\text{model}}(x|\theta) \Big\rangle_{x\sim p_{\text{data}}} \\ &= \sum_{i=1}^{n} \Big\langle -\log p(x_i|\omega^{(i-1)}) \Big\rangle_{x\sim p_{\text{data}}} \; . \end{aligned} \tag{4.11}$$

The successive transformer sampling is illustrated in Fig. 4.4. For each component, $\omega^{(i-1)}$ encodes the dependence on the previous components $x_1, ..., x_{i-1}$, and correspondingly we sample from $p(x_i|\omega^{(i-1)})$. The parameters $\omega^{(0)}, ...\omega^{(i-2)}$ from the sampling of previous components are re-generated in each step, but not used further. This way the event generation is less efficient than the likelihood evaluation during training, because it cannot be parallelized.

Bayesian version

As any generative network, we bayesianize the transformer by drawing its weights from a set of Gaussians $q(\theta)$. In practice, we replace the

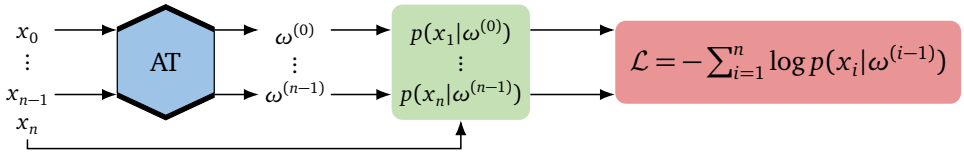

Figure 4.3: Training algorithm for the autoregressive transformer.

deterministic layers of the transformer by Bayesian layers and add the KL-regularization from Eq.(3.6) to the likelihood loss of the transformer, Eq.(4.11)

$$\mathcal{L}_{\text{B-AT}} = \left\langle \mathcal{L}_{\text{AT}} \right\rangle_{\theta\sim q(\theta)} + \text{KL}[q(\theta), p(\theta)]. \tag{4.12}$$

For large generative networks, we encounter the problem that too many Bayesian weights destabilize the network training. While a deterministic network can switch of unused weights by just setting them to zero, a Bayesian network can only set the mean to zero, in which case the Gaussian width will approach the prior $p(\theta)$. This way, excess weights can contribute noise to the training of large networks. This problem can be solved by adjusting the hyperparameter describing the network prior or by only bayesianizing a fraction of the network weights. In both cases it is crucial to confirm that the uncertainty estimate from the network is on a stable plateau. For the transformer we find that the best setup is to only bayesianizing the last layer.

To implement the autoregressive transformer we use PyTorch with the RAdam optimizer. All hyperparameters are given in Tab. A.1. We propose to couple the number of parameters m in the parametrization vector $\omega^{(i-1)}$ to the latent space dimensionality d, because the latent space dimensionality naturally sets the order of magnitude of parameters that the model can predict confidently.

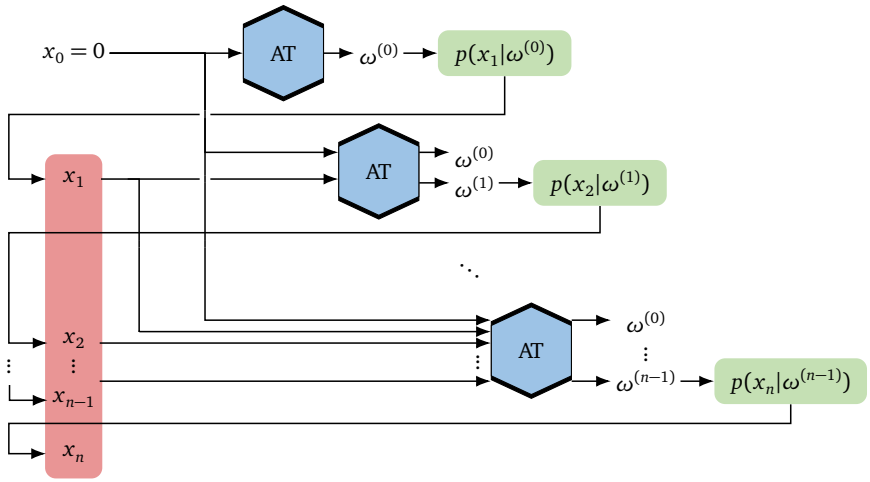

Figure 4.4: Sampling algorithm for the autoregressive transformer.

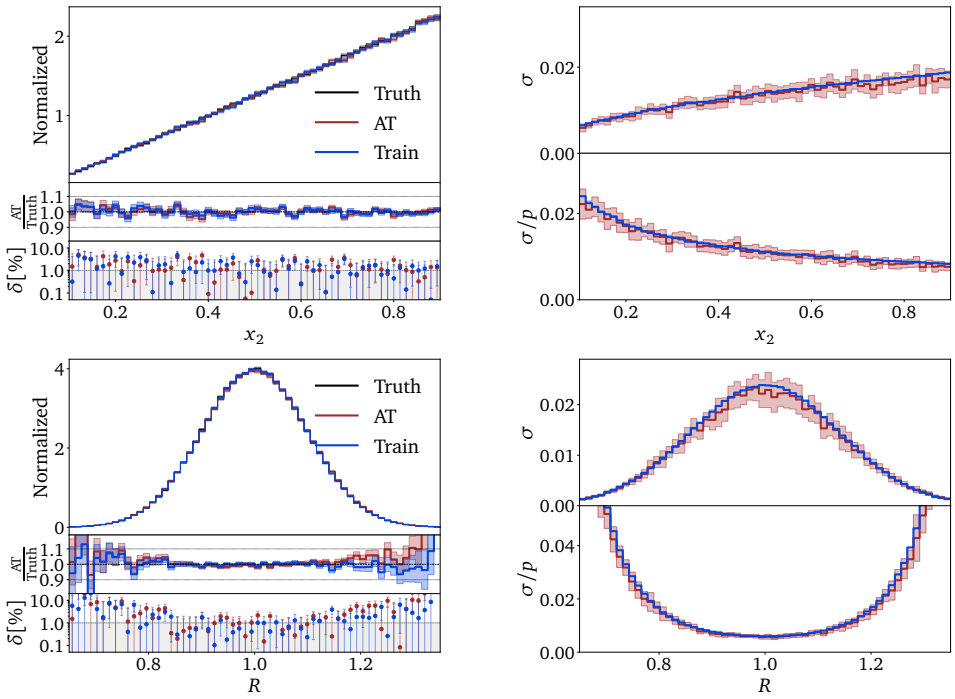

Figure 4.5: Ramp (upper) and Gaussian ring (lower) distribution from the autoregressive transformer with a binned likelihood. We show the learned density and its Bayesian network uncertainty (left) as well as the absolute and relative uncertainties with a range given by 10 independent trainings, compared to the statistical uncertainty of the training data in blue (right).

4.2 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,

$$p_{\text{ramp}}(x_1, x_2) = 2x_2 \, . \tag{4.13}$$

The network input and output are unweighted events. The hyperparameters of the model are given in Tab. A.1. 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 [43, 66]. This way the uncertainty in one dimension is independent of the existence and size of other dimensions.

In Fig. 4.5 we start with a simple representation of the phase space density using 64 bins. In this naive setup the densities of the ramp and the Gaussian ring are described accurately, within our per-cent target range. The largest deviations appear in the tails of the Gaussian ring, but remain almost within the statistical limitations of the training data.

Unlike for the INN and the diffusion models, the uncertainty in the right panels of Fig. 4.5 does not show any real features for the ramp or the Gaussian ring. This shows that the transformer does not use a fit-like density estimation and does not benefit from the increased correlations in the center of phase space [66]. Both of these aspects can be understood from the model setup. First, the autoregressive structure never allows the transformer to see the full phase space density and encode global (symmetry) patterns; second, the main motivation of the transformer is to improve the power-law scaling with the dimensionality of all possible correlations and only focus on the most relevant correlations at the expense of the full phase space coverage.

In Fig. 4.6 we show the same results for a mixture of 21 Gaussians. For this small number of dimensions the advantage over the binned distribution

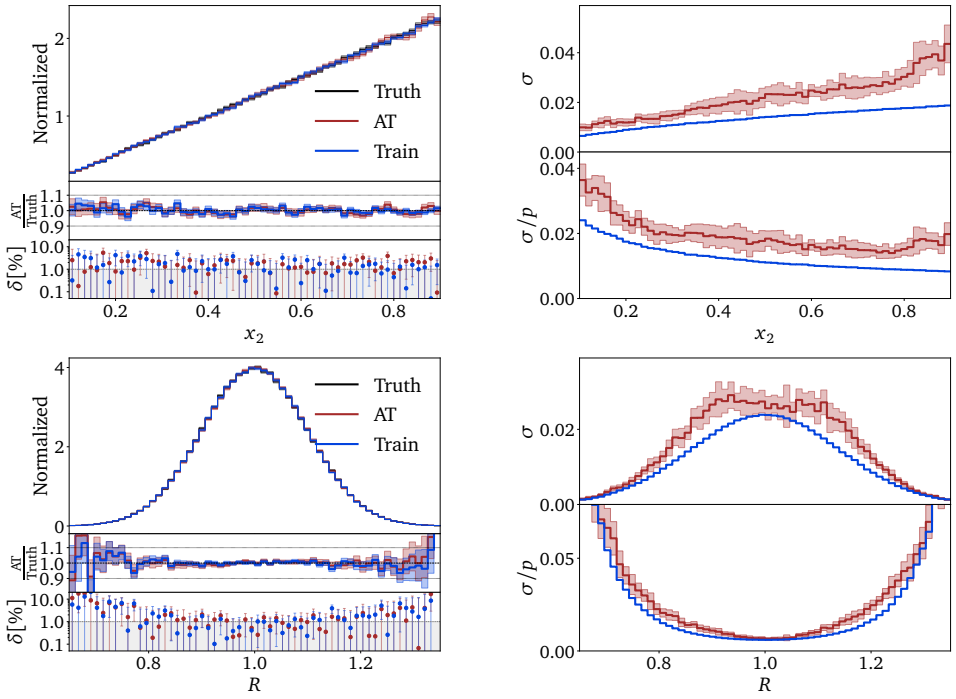

Figure 4.6: Ramp (upper) and Gaussian ring (lower) distribution from the autoregressive transformer with a Gaussian mixture likelihood. We show the learned density and its Bayesian network uncertainty (left) as well as the absolute and relative uncertainties with a range given by 10 independent trainings, compared to the statistical uncertainty of the training data in blue (right).

is not obvious. The main problem appears at the upper end of the ramp, where there exists enough training data to determine a well-suited model, but the poorly-suited GMM just fails to reproduce the flat growth towards the sharp upper edge and introduces a significant artifact, just covered by the uncertainty. For the Gaussian ring the GMM-based transformer is also less precise than the binned version, consistent with the lower resolution in the 2-dimensional model.

The uncertainty predicted by the Bayesian transformer is typically very small. We therefore add the statistical uncertainty of the training data to the right panels of Figs. 4.5 and 4.6, providing a lower bound on the uncertainty. In both cases, the uncertainty of the Bayesian transformer conservatively tracks the statistical uncertainty of the training data.

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

$$p_{\text{ring}}(x_1, x_2) = \mathcal{N}(\sqrt{x_1^2 + x_2^2}; 1, 0.1). \tag{4.14}$$

In Fig. 4.7 we illustrate the unique way in which the GMM-based transformer reconstructs this density. In the left panel, we show $p_{\text{model}}(x_1)$ after the first autoregressive step, constructed out of 21 learned Gaussians. The peaks at ± 1 arise from the marginalization along the longest line of sight. The marginalization also distorts the form of the Gaussians, which are distributed along the ring. The density after the second autoregressive step, $p_{\text{model}}(x_2|x_1)$, is conditioned on the first component. In the second panel we show $p_{\text{model}}(x_2|x_1 = 0)$ with sharp peaks at ± 1 because the event has to be at the edge of the ring. The Gaussians building the left and right peak are distributed roughly equally. On the other hand, $p_{\text{model}}(x_2|x_1 = 1)$ has a broad plateau in the center, again from the x_1-condition.

4.3 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 reconstruc-

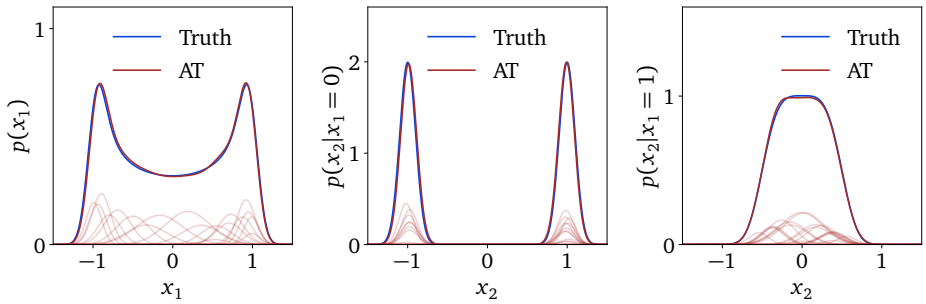

Figure 4.7: Conditional likelihoods for the Gaussian ring. We show the full Gaussian mixture as well as the 21 individual Gaussians, compared to the truth distribution.

tion 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. [43] we choose the the production of leptonically decaying Z-bosons, associated with a variable number of QCD jets,

$$pp \to Z_{\mu\mu} + \{1,2,3\} \text{ jets} . \tag{4.15}$$

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 [11] at 13 TeV, including ISR and parton shower with CKKW merging [67], hadronization, but no pile-up. The jets are defined by FASTJET3.3.4 [68] using the anti-k_T algorithm [69] and applying the basic cuts

$$p_{T,j} > 20 \text{ GeV} \qquad \text{and} \qquad \Delta R_{jj} > 0.4 . \tag{4.16}$$

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

$$\{ p_T, \eta, \phi, m \} . \tag{4.17}$$

Given Eq.(4.16), 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.

We already know from Sec. 4.2 that the autoregressive transformer learns and encodes the phase space density different from normalizing flows or diffusion networks. A key structural difference for generating LHC events is that the transformer can generate events with different jet multiplicities using the same network. The one-hot-encoded jet multiplicity is provided as an additional condition for the training. The autoregressive structure can work with sequences of different length, provided there is an appropriate way of mapping the sequences onto each other. For the LHC events we enhance the sensitivity to the angular jet-jet correlations through the ordering

$$\big((\phi,\eta)_{j_{1,2,3}}, (p_T,\eta)_{\mu_1}, (p_T,\phi,\eta)_{\mu_2}, (p_T,m)_{j_{1,2,3}} \big) . \tag{4.18}$$

While the Bayesian transformer does learn the angular correlations also when they appear at the end of the sequence, this ordering provides a

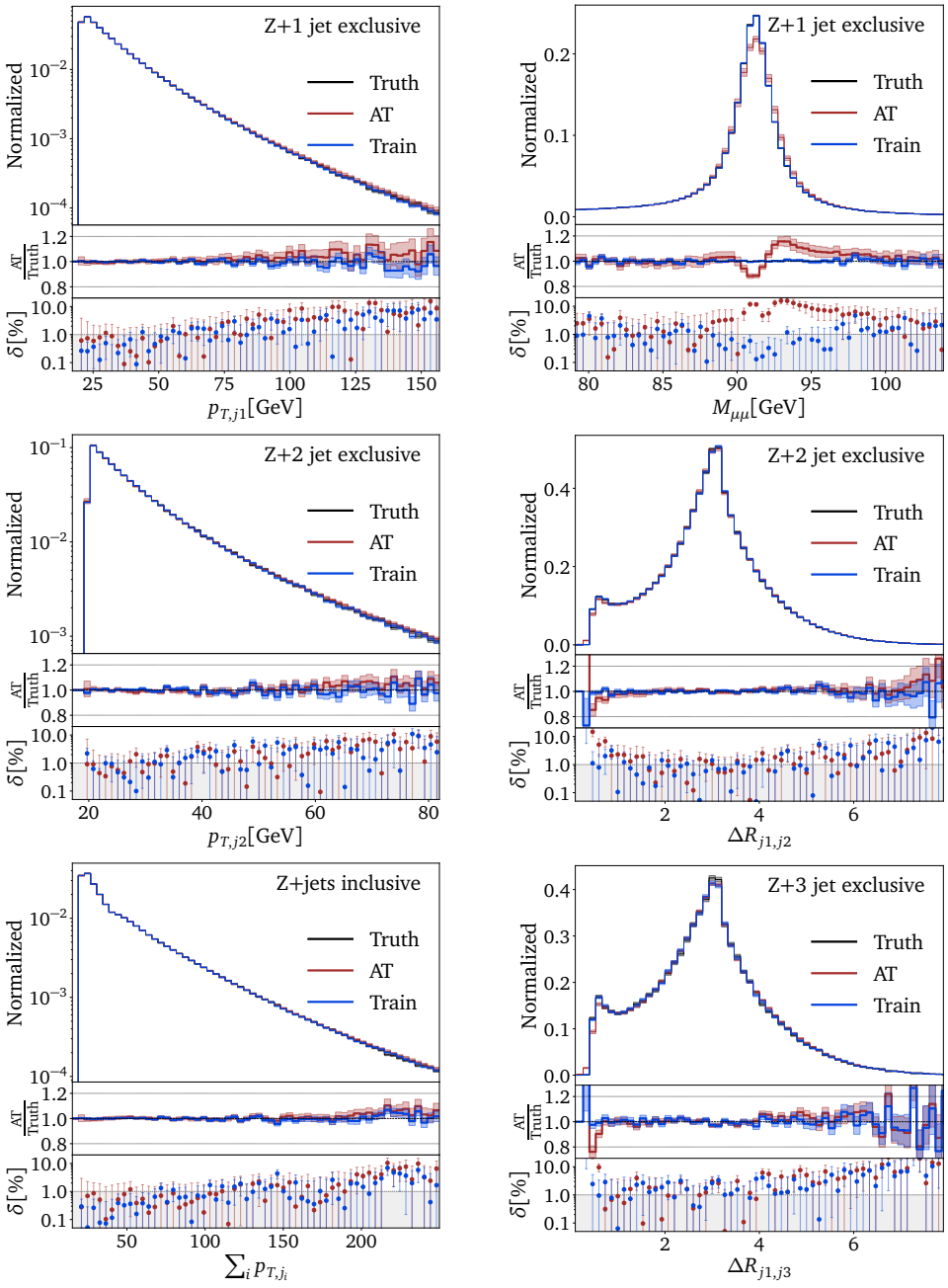

Figure 4.8: Bayesian autoregressive transformer densities and uncertainties for $Z+1$ jet (upper), $Z+2$ jets (center), and $Z+3$ jets (lower) from combined $Z+$ jets generation. The uncertainty on the training data is given by bin-wise Poisson statistics. The network architecture is given in Tab. A.1. For a comparison with the INN we refer to Fig. 11 of Ref. [43].

significant boost to the network's precision. For the transformer training, we want the features of the 3-jet to be well represented in the set of vectors defined in Eq.(4.18). To train on equal numbers of events with one, two, and three jets, we sample 1-jet and 2-jet events randomly at the beginning of each epoch. The loss is first evaluated separately for each jet multiplicity, and then averaged for the training update.

In Fig. 4.8 we show a set of kinematic distributions for different jet multiplicities, including the jet-inclusive scalar sum of the up to three $p_{T,j}$. 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 AT-generated events. 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-AT. Sampling a variable number of jets with the multiplicity as a condition leads to no additional complication.

In the right panels of Fig. 4.8 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 AT 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. The benefit of this ordering (4.18) can be seen in these distributions, which are reproduced at the per-cent precision without any additional effort. This is true for $\Delta R_{j_1 j_2}$ and $\Delta R_{j_1 j_3}$, reflecting the democratic ordering and training dataset. The sharp phase space boundary at $\Delta R_{jj} = 0.4$ can be trivially enforced during event generation.

4.4 OUTLOOK

Generative neural networks are revolutionizing many aspects of our lives, and LHC physics is no exception. Driven by large datasets and precise first-principle simulations, LHC physics offers a wealth of opportunities for modern machine learning, in particular generative networks [70]. Here,

classic network architectures have largely been surpassed by normalizing flows, especially its INN variant, but cutting-edge new approaches are extremely promising. In this study we focus on autoregressive generative transformers.

We have first implemented a Bayesian network version, which allows us to understand the way that the network approaches the density estimation. While INNs and diffusion networks first identify classes of functions and then adapt them to the correlations in phase space [66], the transformer learns patterns patch-wise and dimension by dimension.

Next, we have applied the autoregressive transformer 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 [43]. The most challenging phase space correlations are the narrow Z-peak and the angular jet–jet separation combined with a collinear enhancement.

Altogether, we have found that new generative network architectures have the potential to outperform even advanced normalizing flows and INNs. However, autoregressive transformers 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 one of the new architectures.

5

EXTRAPOLATING JET RADIATION WITH AUTOREGRESSIVE TRANSFORMERS

The research presented in this chapter is based on work in collaboration with Anja Butter, François Charton, Javier Mariño Villadamigo, Ayodele Ore and Tilman Plehn, and has been previously published in Ref. [71]. All tables and figures as well as parts of the text are similar or identical to the content of these articles.

In this chapter we tackle the physics problem of using generative networks to describe jets radiated from a hard scattering process. In fundamental QCD, jet radiation is described by successive probabilistic parton splittings. It is an integral part of QCD predictions for hadron colliders, where final states with a fixed number of jets are not in line with parton densities and collinear factorization [72–74]. The corresponding splitting kernels and the generated phase space correlations are approximately universal [75]. The generated number of jets follows well-defined patterns, also predicted by QCD.

Autoregressive generative networks can, just like with language, generate open-end sequences of particles, or events with a variable number of particles. An autoregressive generation requires a factorized phase space probability[76, 77]. This structure matches the QCD aspects of universal splittings and well-defined jet numbers. Our generative architecture of choice is an autoregressive transformer [8, 78].

The goal of this chapter is to show, that a generative transformer can extrapolate in the number of jets and generate approximately universal jet radiation for higher jet numbers than seen during the training. In Sec. 5.1 we describe the QCD structures motivating an approximately factorized phase space likelihood and its ML-realization, leading to our autoregressive generative transformer. We then present extrapolated predictions in the number of jets in Sec. 5.2, using bootstrapped training data in Sec. 5.2.2, a truncated loss without fixed stopping condition in Sec. 5.2.3, and a loss that overrides the stop condition in Sec. 5.2.4. In Appendix A.2 we provide additional information on how to improve the accuracy of the generative transformer through including a classifier in the training, in the spirit of a GAN.

5.1 AUTOREGRESSIVE JET RADIATION

Given that jet radiation in QCD is described by universal splitting kernels and well-defined scalings in the number of jets, we will train an autoregressive transformer with a factorized likelihood loss to generate QCD jet radiation. The ultimate goal is to show that the transformer not only

describes jet radiation to a number of jets represented in the training data, but that it can extrapolate to larger jet counts than seen during training.

We first remind ourselves of universal splittings in QCD and the typical scaling in the number of produced jets. We will then motivate our Z+jets dataset, exhibiting the universal so-called staircase scaling. To train a generative network we first derive a factorized phase space probability and then encode it in a loss function for an autoregressive transformer.

5.1.1 *QCD jet radiation*

Collinear parton splittings in the initial or final states are the backbone of QCD predictions for hadron colliders. Their universal nature is the basis of parton densities, parton showers, and jet radiation, and it defines the structure of LHC events [72–74]. A challenging consequence of collinear splittings is that any hard scattering process is accompanied by a variable number of jets in the final state, as described by jet radiation and parton showers in the multi-purpose event generators [10, 79–81]. Combining parton shower and hard matrix element predictions is the theory basis for the entire precision physics program at the LHC [67, 82–84].

Universal autoregressive structure

The physics background of this chapter is the universal nature of jet radiation from collinear splittings, reflecting the collinear factorization of the matrix element and the phase space. It allows us to generate events with $n+1$ final-state jets from events with n final-state jets. For final state radiation this factorization is schematically written as

$$\sigma_{n+1} \sim \int \frac{dp^2}{p^2} dz \, \frac{\alpha_s}{2\pi} P(z) \sigma_n \, , \tag{5.1}$$

where p^2 is the invariant mass of the splitting parton, z is the momentum fraction carried out of the hard process σ_n, and $P(z)$ are the universal collinear splitting kernels. In the initial state, this factorization is the basis of the DGLAP equation with the subtracted versions of the same collinear splitting kernels.

The iterative structure of Eq.(5.1) allows us to simulate parton splittings as Markov processes, and it also allows us to describe the underlying densities in an approximately factorized form. Such a factorized density is most efficiently generated by an autoregressive structure. The key ingredients are the perturbative QCD splitting functions and the non-splitting probability, referred to as Sudakov factor.

The actual simulation of, approximately, collinear jet radiation is not expected to be exact: first, we need to generate final transverse momenta for the radiated partons while keeping transverse momentum conservation [85]; second, we need to correct for color and spin correlations [86]; finally, the structure of successive $(1 \to 2)$-splittings might be not be sufficient for the LHC precision [87, 88]. Nevertheless, the form of Eq.(5.1)

suggests that in QCD events with increasing number of jets can be derived from a simple iterative pattern, and such a pattern can in principle be learned and extrapolated by a neural network with the right (autoregressive) architecture.

Jet rate scaling

Unfortunately, the number of radiated jets in LHC events does not follow a universal distribution. However, we can derive two distinct patterns. Both are defined in terms of the ratio of $(n+1)$-jet to n-jet events or in terms of the fraction of events with n jets,

$$R_{(n+1)/n} = \frac{\sigma_{n+1}}{\sigma_n} \quad \text{and} \quad P(n) = \frac{\sigma_n}{\sigma_{\text{tot}}} \quad \text{with} \quad \sigma_{\text{tot}} = \sum_{n=0}^{\infty} \sigma_n \,. \tag{5.2}$$

The ratios and the probabilities depend on kinematic cuts regularizing the soft and collinear divergences, typically the minimum transverse momentum of the counted jets, $p_{T,\text{min}}$.

1. The first pattern, Poisson scaling, implies in terms of the expectation value $\bar{n}$,

$$R_{(n+1)/n} = \frac{\bar{n}}{n+1} \qquad \Leftrightarrow \qquad P(n) = \frac{\bar{n}^n e^{-\bar{n}}}{n!} \,. \tag{5.3}$$

   At colliders, it occurs for processes with large splitting probabilities and large scale differences, for instance multi-jet production in e^+e^- collisions.

2. We focus on the alternative staircase scaling [89–91] with

$$R_{(n+1)/n} = e^{-b} \qquad \Leftrightarrow \qquad P(n \geqslant n_{\text{min}}) = e^{-b n_{\text{min}}} \,. \tag{5.4}$$

   While the ratio e^{-b} is the same for the exclusive and inclusive jet counts, the probability only has a simple form for the inclusive jet count, classifying events with n_{min} jets or more. We can use the universal scaling to relate $P(n)$ to a successive or conditional probability

$$P(n+1|n) = R_{(n+1)/n} \,. \tag{5.5}$$

At colliders, staircase scaling is predicted for smaller splitting probabilities and democratic scales [92]. In that case, the jet count distributions can be derived from QCD using generating functionals [93]. For final state radiation we quote the scale-dependent result

$$R_{(n+1)/n} = 1 - \tilde{\Delta}_g(Q^2) \,, \tag{5.6}$$

with a modified Sudakov factor or non-splitting probability

$$\tilde{\Delta}_g(Q^2) = \exp\left[-C_A \int_{Q_0^2}^{Q^2} dt \frac{\alpha_s(t)}{2\pi t} \left(\log \frac{t}{Q_0^2} - \frac{11}{6}\right)\right] \,. \tag{5.7}$$

To leading-log level the integrand is the QCD splitting function in the collinear approximation. This QCD derivation of staircase scaling requires democratic scales $Q^2/Q_0^2 \sim \mathcal{O}(1)$.

At the LHC the standard example is weak boson production with jets,

$$pp \to Z + n \text{ jets} \qquad \text{with} \qquad n = \{0, 1, 2, 3, ...\} \tag{5.8}$$

Because the two scaling patterns are different, we will limit ourselves to learning and generating staircase scaling from datasets described by universal collinear radiation.

5.1.2 *Z + jets dataset*

We follow the above motivation and Refs. [8, 43] by generating leptonically decaying Z bosons in association with a variable number of jets. Unlike for earlier studies, we include higher jet multiplicities to provide a challenge for the transformer

$$pp \to Z_{\mu\mu} + \{0, \cdots, 10\} \text{ jets}. \tag{5.9}$$

We use MadGraph5_aMC_@NLOv3.5.1 to generate 500M events, including ISR and parton shower with Pythiav6.3, using CKKW merging and hadronization, but no pile-up. The jets are defined with FastJetv3.3.4 using the anti-k_T algorithm with the basic cuts

$$p_{T,j} > 20 \text{ GeV} \qquad \text{and} \qquad \Delta R_{jj} > 0.4. \tag{5.10}$$

The muons and jets are both ordered in transverse momentum. Our phase space dimensionality is three per muon and four per jet. Momentum conservation is not guaranteed, because some final-state particles might escape for instance the jet algorithm. The distribution of the number of jets and the corresponding ratios $R_{(n+1)/n}$ are shown in the two panels of Fig. 5.1. We observe an approximately constant ratio for most of the spectrum, confirming a staircase scaling as defined in Eq.(5.4). Towards large numbers of jets we start encountering statistical limitations as well as phase space limitations.

Of our 500M events we use 80% for training, 10% for validation, and 10% for testing. The number of events per jet multiplicity is given in Tab. 5.1. To avoid being entirely dominated by low-multiplicity events, we cap the number of events with $n = 0, 1, 2$ to match the number of events with $n = 3$.

For the jet momenta, we use a minimal preprocessing [8, 43], where each particle i is represented in standard jet coordinates

$$\{ (p_T, \eta, \phi, m)_i \} . \tag{5.11}$$

We enforce the p_T cuts in Eq.(5.10) using the transformation $\log(p_T - p_{T,\text{min}})$, which maps allowed transverse momenta to the full real line and leads to an approximately Gaussian shape. The jet mass is encoded as

Number of jets	0	1	2	3	4	5	6	7	8	9	10
Number of events	380M	91M	21M	4.7M	1.1M	230k	52k	11k	2.3k	510	95
Cap	4.7M	4.7M	4.7M	-	-	-	-	-	-	-	-

Table 5.1: Event counts for our simulated Z+jets dataset. When training networks, we cap the size of the 0,1,2-jet subsets.

$\log m$. We express angles ϕ relative to the leading muon and apply a special treatment described in Section 5.1.4 to reflect the periodicity. Finally, we standardize all phase space variables except ϕ as $(x - \bar{x})/\sigma(x)$. For 10 jets the phase space is 45-dimensional.

5.1.3 *Factorized probability*

Following the discussion in Sec. 5.1.1, QCD jet radiation has two features that make it an attractive target for autoregressive generative networks: the universal splitting kernels and the jet ratio patterns. In case of staircase scaling the ratios of exclusive and inclusive jet rates are also universal. Equation (5.1) suggests that the phase space density for an event x can be constructed as a product of conditional distributions, each taking the form

$$p(x_i|x_{1:i-1}) = p_{\text{kin}}(x_i|x_{1:i-1})\, p_{\text{split}}(x_{1:i-1}) \,, \tag{5.12}$$

where we denote by $x_{1:i-1}$ the sequence of particles $x_1, \ldots, x_{i-1}$. For particle i, p_{kin} encodes the kinematics, conditional on the probability p_{split} that it will be radiated. Both probabilities are conditioned on the full previous sequence of particles $x_{1:i-1}$.

Approximate universality of the splitting kernels and jet ratios translates to universality of p_{kin} and p_{split} respectively. This raises the possibility that, given the right architecture, we can train a neural network to extrapolate QCD jet radiation patterns in analogy to a collinear parton shower Monte Carlo approach.

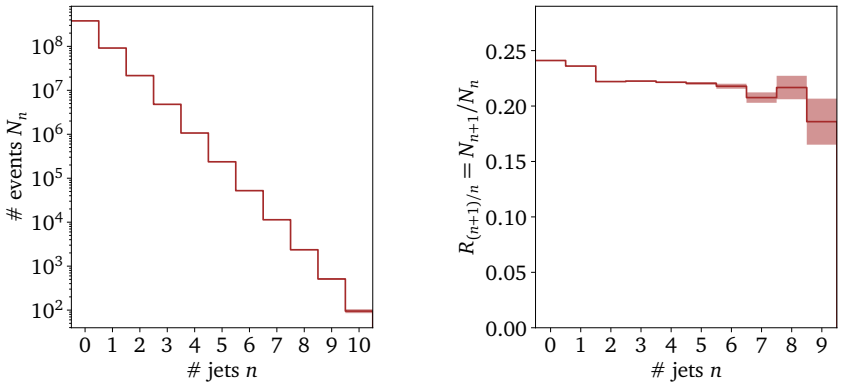

Figure 5.1: Staircase scaling of the number of jets in our $pp \to Z + n$ jets dataset. We show statistical uncertainties and use Gaussian error propagation to estimate the uncertainties for the ratio $R_{n+1/n}$.

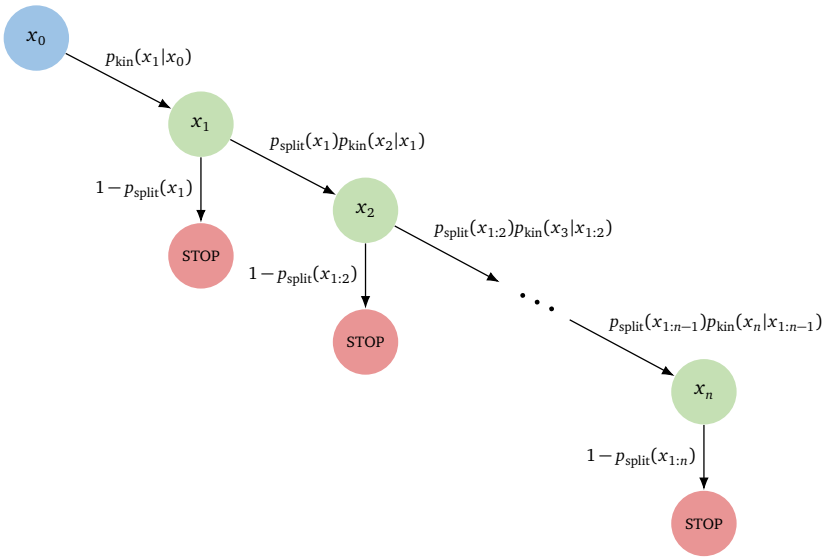

Figure 5.2: Probability tree for variable-length event generation. To disallow empty events, we assign $p_{\text{split}}(x_0) = 1$.

Using the conditional probabilities in Eq.(5.12) we can build the likelihood of an n-jet event,

$$\begin{aligned} p(x_{1:n}) &= \left[\prod_{i=1}^{n} p(x_i|x_{1:i-1})\right] \left[1 - p_{\text{split}}(x_{1:n})\right] \\ &= \left[\prod_{i=1}^{n} p_{\text{kin}}(x_i|x_{1:i-1})\right] \left[\prod_{i=1}^{n} p_{\text{split}}(x_{1:i-1})\right] \left[1 - p_{\text{split}}(x_{1:n})\right] , \end{aligned} \tag{5.13}$$

where the last term gives the probability that there are no further splittings and the event is complete. In QCD language it corresponds to a Sudakov factor. In accordance with Eq.(5.11), the phase space probability $p(x_{1:n})$ has a well-defined dimensionality 4^n. It is normalized both as a continuous distribution over x_i and a categorical distribution over n,

$$\sum_{n=1}^{\infty} \int dx_1 \dots dx_n \, p(x_{1:n}) = 1 \, . \tag{5.14}$$

As illustrated in Fig. 5.2, the generative process can be visualized as a binary probability tree with a Sudakov stop if no further splitting happens. The combination of p_{split} for a splitting or $(1 - p_{\text{split}})$ for no splitting is described by a Bernoulli distribution p_{bin} with expected splitting probability p_{split}

$$p_{\text{bin}}\left(y|p_{\text{split}}\right) = p_{\text{split}}^{y}(1 - p_{\text{split}})^{1-y} \quad \text{with} \quad y \in \{0, 1\},\ p_{\text{split}} \in [0, 1] \, . \tag{5.15}$$

It allows us to unify the factors p_{split} and $1 - p_{\text{split}}$ in a completely factorized likelihood

$$p(x_{1:n}) = \prod_{i=1}^{n} p_{\text{kin}}(x_i|x_{1:i-1}) \prod_{i=0}^{n} p_{\text{bin}}\big(1 - \delta_{in}|p_{\text{split}}(x_{1:i})\big) \,. \tag{5.16}$$

The Kronecker delta assigns the splitting label zero for the n^{th} particle and one otherwise. By keeping the full conditioning on $x_{1:i}$, this likelihood is completely general and can capture non-universal correlations. This is important when we describe full events, including the hard process. For $Z_{\mu\mu}$+jets events, we also treat the muons autoregressively and enforce a splitting probability of one for them.

Similarly to the autoregressive decomposition of the likelihood of different particles $p(x_{1:n})$, we factorize the likelihood of individual particles $p_{\text{kin}}(x_{i+1}|x_{1:i})$ in terms of their components. The ordering of components can affect the network performance [8], however for such small sequences this effect is negligible. The elements of the sequence are one-dimensional, and we parametrize their distributions with mixtures

$$\begin{aligned} p_{\text{kin}}(x_{i+1}|x_{1:i}) = {} & p_{\text{GM}}(p_{T,i+1}|x_{1:i})\, p_{\text{vMM}}(\phi_{i+1}|x_{1:i}, p_{T,i+1}) \\ & \times p_{\text{GM}}(\eta_{i+1}|x_{1:i}, p_{T,i+1}, \phi_{i+1}) \\ & \times p_{\text{GM}}(m_{i+1}|x_{1:i}, p_{T,i+1}, \phi_{i+1}, \eta_{i+1}) \,. \end{aligned} \tag{5.17}$$

We use Gaussian mixtures p_{GM} for non-periodic variables and von Mises mixtures p_{vMM} for the periodic variable ϕ. We do not generate the fixed muons mass in $Z_{\mu\mu}$+jets events. Periodic likelihoods for angular variables inform the network about this geometric information and therefore improve the performance. This has been previously shown for normalizing flows [94] and conditional flow matching [95].

In contrast to the autoregressive structure of $p(x_{1:n})$ in Eq.(5.13), Eq.(5.18) is not inspired by physics and other choices are possible. Examples from the literature are categorical distributions over bins (which suffer from limited resolution) [8, 78, 96], normalizing flows [35, 97], and conditional flow matching [35, 97].

We emphasize that this factorized likelihood, built to describe an autoregressive generation, generalizes the usual factorization $p(x_{1:n}) = p(x_{1:i}|i)p(i)$ [35, 43] and previous autoregressive approaches [8]. Similar generative approaches have been developed for jet constituent generation [98], and a similar factorization for density estimation has been studied in Refs. [76, 77].

5.1.4 *Autoregressive transformer*

Starting from the physics-motivated factorization in Eq.(5.16), we need to encode these densities with variable-length inputs $x_{1:i}$ using neural networks. A transformer f_θ with causal attention mask will turn these

sequences into fixed-sized representations. We use a pre-layernorm transformer decoder with GeLU activations, for more information see App. A.2, and decompose the transformer output as

$$f_\theta(x_{1:i}) = (\rho_i, \nu_i) \in \mathbb{R} \times \mathbb{R}^d \, . \tag{5.18}$$

The embedding dimension d is a hyperparameter. The ρ_i represent the splitting probabilities that parametrize the Bernoulli distributions,

$$\rho_i \approx p_{\text{split}}(x_{1:i}) \, . \tag{5.19}$$

The embeddings ν_i similarly parametrize the kinematic conditionals

$$p_{\text{kin}}(x_i|\nu_{i-1}) \approx p_{\text{kin}}(x_i|x_{1:i-1}) \, . \tag{5.20}$$

For clarity, we always suppress the dependence of ρ_i and ν_i on $x_{1:i}$ and on the transformer parameters θ.

Loss and training

The loss function of the autoregressive network is given by the likelihood in Eq.(5.16),

$$\begin{aligned} \mathcal{L}_{\text{like}} &= -\Big\langle \log p(x_{1:n}) \Big\rangle_{x\sim p_{\text{data}}} \\ &= -\Big\langle \sum_{i=1}^{n} \log p_{\text{kin}}(x_i|\nu_{i-1}) + \sum_{i=0}^{n} \log p_{\text{bin}}(1-\delta_{in};\rho_i) \Big\rangle_{x\sim p_{\text{data}}} \, . \end{aligned} \tag{5.21}$$

The first term is the usual likelihood loss for the kinematic generative network. The second term is the standard binary cross entropy. In our generative network it implicitly enforces the correct event multiplicity through a splitting discriminator.

In Sec. 5.2 we will consider modified training strategies to extrapolate beyond the maximal multiplicity n_{max} of events contained in the training dataset. One strategy is to modify the cross entropy part of the likelihood loss in Eq.(5.21), for example by removing the contribution from the term with highest multiplicity n_{max}

$$\begin{aligned} \mathcal{L}_{\text{trunc}} = \Big\langle & -\sum_{i=1}^{n} \log p_{\text{kin}}(x_i|\nu_{i-1}) \\ & -\sum_{i=0}^{n} (1-\delta_{in_{\text{max}}}) \log p_{\text{bin}}(1-\delta_{in};\rho_i) \Big\rangle_{x\sim p_{\text{data}}} \, . \end{aligned} \tag{5.22}$$

Using this loss, the splitting prediction for maximum-multiplicity events, $\rho_{n_{\text{max}}}$, is not explicitly trained. Rather, the weight sharing in the transformer allows correlations learned at lower multiplicity to be recycled.

When training our transformers on the Z+jets dataset from Sec. 5.1.2, we use the Adam optimizer with constant learning rate 3×10^{-4} and batch

size 1024. The batches contain events with different multiplicities following the distribution in the training data. The validation loss is tracked every 5k iterations, and we restore the network from the checkpoint with lowest validation loss after 200k iterations.

Sampling

To generate full events x, we sequentially sample from the likelihood described in Sec. 5.1.3, as visualized in Fig. 5.2. We sample 10M events in total and split them according to their multiplicities. This procedure generates samples from the exact likelihood learned by the network, but does not give us explicit control over the generated jet multiplicities. We decide on a maximum number of jets, and discard events for which the transformer predicts further splittings.

Bayesian network

Because we hope to use the autoregressive transformer for extrapolation beyond the jets present in the training data, we need to quantify the uncertainty in the predicted phase space density. We resort to Bayesian neural networks (BNN) [53–56] as a way to learn systematic and statistical uncertainties together with the mean network predictions. These are a standard method in LHC physics, for instance for amplitude regression [39], calibration [99, 100], and classification [101]. They can be generalized to the density estimation aspect of generative networks [8, 43, 66, 102], where they return an uncertainty on the unit event weight.

BNNs replace the network parameters θ by learnable distributions $q(\theta)$, usually assumed to be uncorrelated Gaussians. Their loss consists of a sampled likelihood term and a regularization with a prior-width hyperparameter,

$$\mathcal{L}_{\text{BNN}} = -\Big\langle \log p(x) \Big\rangle_{x\sim p_{\text{data}},\theta\sim q} + D_{\text{KL}}\left[q(\theta), p(\theta)\right] \; . \tag{5.23}$$

To evaluate the BNN we sample from the learned weight distributions, in our case generating 10 samples, with a given number of 10M events each.

5.2 RESULTS

Even though we are interested in extrapolating towards unseen jet numbers, we first benchmark the accuracy of our transformer in Sec. 5.2.1. We also show how without modifications the generative network does not actually extrapolate. For a successful extrapolation we first use a bootstrap approach in Sec. 5.2.2 and then show in Sec. 5.2.3 and Sec. 5.2.4 how truncating or overriding the likelihood loss allows the network to generate larger jet numbers than seen during training.

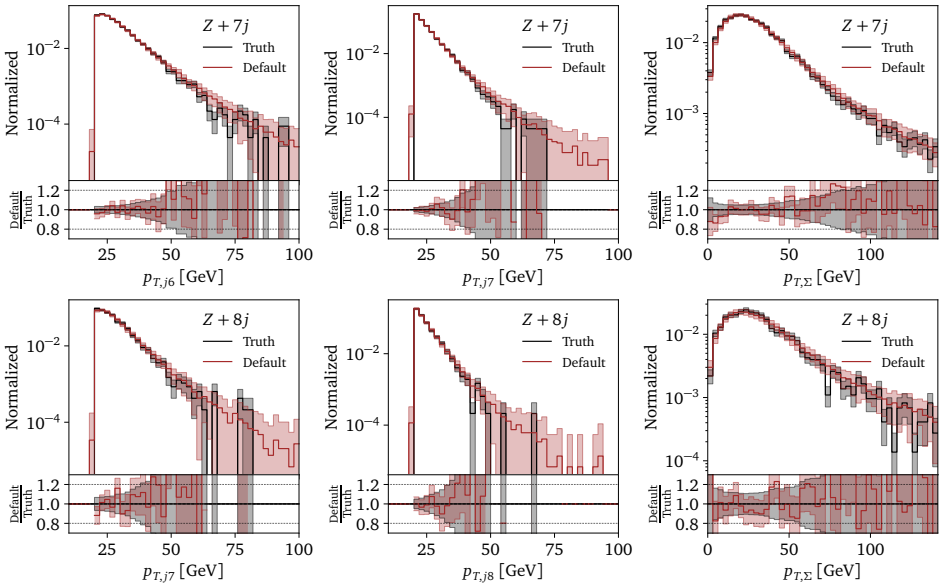

Figure 5.3: Selection of features in Z + 7 and 8-jet events for the generative network trained on the full dataset, including 7 and 8-jet events.

5.2.1 *Generating without extrapolation*

We begin by demonstrating that our transformer learns the phase space density precisely across event multiplicity. We train a Bayesian version of the transformer using all $Z + n$ jet events, from the hard process only, or $n = 0$, up to $n = 10$. We sample 10M events each from 10 BNN predictions. The jet multiplicity distribution is shown in Fig. 5.4, showcasing that the generator can learn the universal staircase scaling. In Fig. 5.3, we see that the network reproduces the kinematic distributions with precision down to the statistical uncertainty of the test set. The transverse component of the vector sum of all particle momenta, $p_{T,\Sigma}$, provides a sensitive test of

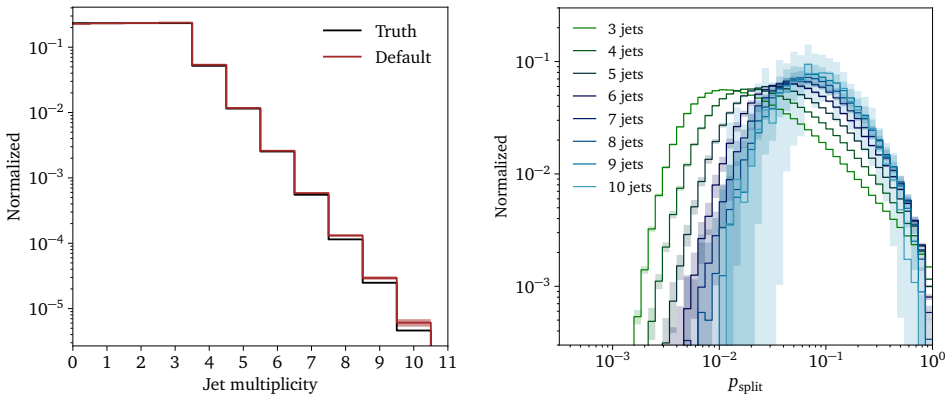

Figure 5.4: Jet multiplicity distribution (left) and splitting probabilities (right) for samples generated with the transformer trained on the full dataset up to 10 jet events.

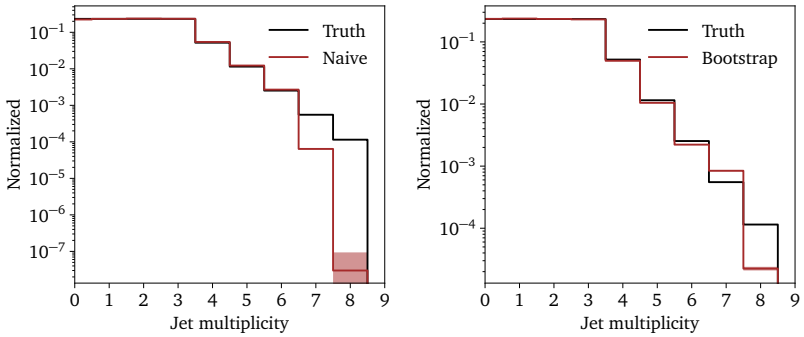

Figure 5.5: Jet multiplicity distributions for samples generated with the transformer using the naive training (left) and bootstrapping (right).

learned global correlation among all particles. All deviations from the training data are captured by the Bayesian uncertainty.

Next, we inspect whether the network has learned universal structure in the probability to generate additional jets. In Fig. 5.4 we show the distributions of p_{split} predicted during the autoregressive sampling steps. We train the network on the entire dataset, with up to 10 jets. We ignore the learned p_{split} for the first two jets, because we manually capped the number of training events for up to two jets, as shown in Tab. 5.1. For more than 6 jets, the distribution stabilizes within the Bayesian uncertainty band, indicating that between 6 and 10 jets we do not observe a significant effect from the parton densities [93].

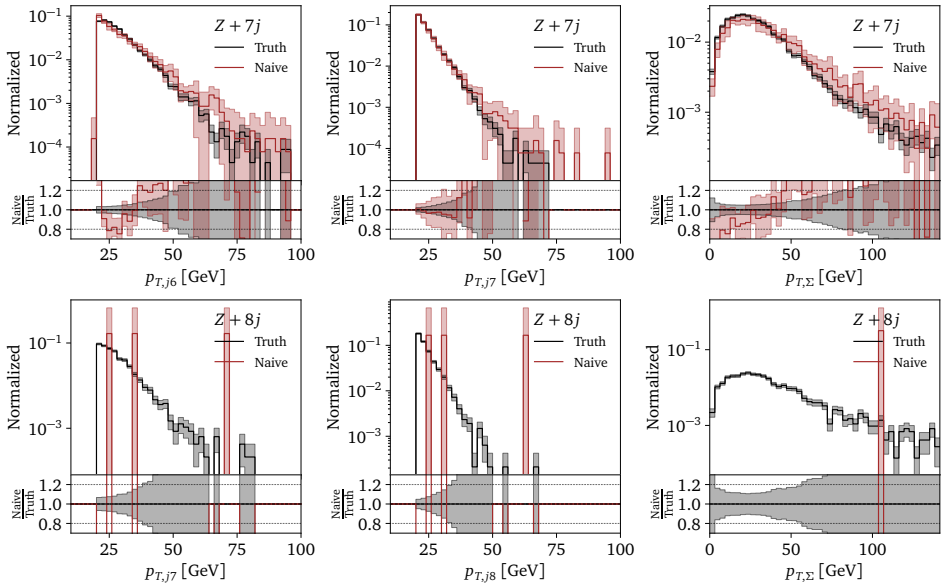

Figure 5.6: Selection of features in Z + 7 and 8-jet events for a generator trained on up to 6-jet events.

Naive extrapolation

Because the termination of the number of jets is implemented probabilistically, we can naively extrapolate to higher jet numbers. For instance, we can train the networks with up to 6 jets and assess the small number of 7-jet and 8-jet events they generate. While the quality of 7-jet and 8-jet events should be worse than for jet numbers seen during training, we want to know if the transformer can leverage universal properties of the QCD jet radiation. We show the generated jet multiplicity distribution in the left panel of Fig. 5.5. Indeed, the network generates events with more than 6 jets, albeit with much lower probability than expected from staircase scaling.

For perfect training, we expect the rate for events with more jets than the training set to approach zero. This is because the transformer output ρ_i is trained to match the probability that another jet follows particle i, $\rho_i \approx p_{\text{split}}(x_{1:i})$. In a given training set, with maximum event length n_{max}, one always has

$$p_{\text{split}}(x_{1:n_{\text{max}}}) = 0 \,. \tag{5.24}$$

The optimal network would learn $\rho_{n_{\text{max}}} = 0$, and the transformer can ignore physical correlations. The reason we do not observe exact zero splitting probabilities is that the weight sharing in the autoregressive transformer imparts a bias to reuse the pattern learned at low multiplicities.

Given the small but finite rate of 7-jet events generated through naive extrapolation, we want to see if the transformer has generalized the jet kinematics. In Fig. 5.6, we show the kinematic features as for the extrapolated 7-jet events. Among 100M generated events, the transformer only generates three 8-jet events, so we cannot assess their quality. However, the 7-jet events look qualitatively reasonable. In particular, the slightly broken transverse momentum conservation is reproduced with an accuracy similar to the baseline in Fig. 5.3. Given that the p_T of the 7^{th} jet is approximately the same scale as the level of momentum non-conservation, this is a non-trivial result. It suggests that the transformer indeed generalizes kinematics, and we should mainly address the learned jet multiplicity.

5.2.2 *Extrapolation with bootstrap*

A simple modification to increase the fraction of learned 7-jet events is to bootstrap them, i.e. add generated 7-jet events to the training data. This way, we dynamically break the condition $p_{\text{split}}(x_{1:n_{\text{max}}}) = 0$ of Eq.(5.24) and allow the network to adapt its multiplicity distribution. By repeating this bootstrapping, we can also generate 8 jets and beyond. The fraction of generated events introduced to the training dataset is a hyperparameter. It controls the learned multiplicity distribution.

We start to add bootstrapped events after a warm-up stage of 200k iterations, corresponding to a full-length training in the appraoch of Sec. 5.2.1.

Without this warm-up stage, the network memorizes the poor-quality samples of the freshly initialized network. After the warm-up, we generate a buffer of 1k 7-jet events. For every generated batch we sample a new deterministic network from the learned weight distribution, making sure that we cover the full range of the weight posterior distribution. We then add a single 7-jet event to each batch of 1024 events, corresponding roughly to the fraction of 7-jet events in the training dataset, and train for another 200k iterations with these settings. After every 50 iterations, we generate a batch of 32768 events, extract the 7-jet events and add them to the buffer. Once the buffer contains 50k events, we start to replace its oldest events with newly generated events. This allows the network to dynamically adapt the quality of 7-jet events. We observe that the network has to be trained for a sufficient amount of time in the bootstrapping mode to adapt to the changed multiplicity distribution.

The obtained jet multiplicity distribution is shown in the right panel of Fig. 5.5. We now get significantly more 7-jet and 8-jet events, indicating that the network indeed adapts the multiplicity distribution. The fraction of 8 jet events is significantly lower than in the training data, because we only bootstrap 7 jet events. The kinematical distributions of the generated events are shown in Fig. 5.7. They show that the bootstrapping generator yields valid kinematic configurations. However, there are deviations in the kinematic features from the truth that are not covered by the Bayesian uncertainty.

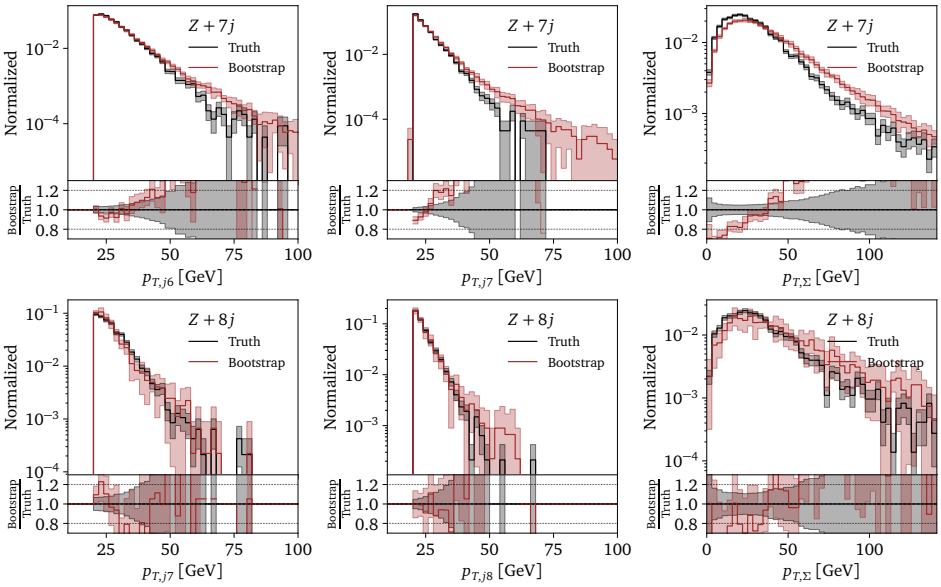

Figure 5.7: Selection of features in Z + 7 and 8-jet events for a generator trained on up to 6-jet events using the bootstrap technique.

5.2.3 *Extrapolation with truncated loss*

A complementary way to combat the suppression of events with more jets than the training set is to modify the likelihood loss. As discussed in Sec. 5.2.1, the cause of the suppression is the constant $p_{\text{split}}(x_{1:n_{\text{max}}}) = 0$ represented by a training dataset with at most n_{max} jets. A simple solution is to omit the final Bernoulli contribution from the loss and truncate the loss as described in Eq.(5.22),

$$\mathcal{L}_{\text{trunc}} = \Big\langle -\sum_{i=1}^{n} \log p_{\text{kin}}(x_i|v_{i-1}) - \sum_{i=0}^{n} (1-\delta_{in_{\text{max}}}) \log p_{\text{bin}}(1-\delta_{in};\rho_i) \Big\rangle_{x\sim p_{\text{data}}}, \tag{5.25}$$

It differs from the complete likelihood loss of Eq.(5.21) in the addition of the factor $1-\delta_{in_{\text{max}}}$ in front of the Bernoulli component. Now, the splitting prediction for maximum-length events, $\rho_{n_{\text{max}}}$, is not explicitly trained. Rather, the weight sharing in the transformer allows correlations learned at lower multiplicity to be recycled. When sampling a network trained in this way, the splitting predictions beyond n_{max} are pure extrapolation.

Using the truncated loss, we again train a transformer on events with up to 6 jets and again sample up to 8 jets. The generated multiplicities are shown in Fig. 5.8. Indeed, the network learns and extrapolates the staircase scaling. We show the extrapolated kinematic correlations in Fig. 5.9. The only deviation exceeding the BNN uncertainty is a slightly larger transverse momentum imbalance than expected in 7-jet events. This result demonstrates that the generative transformer described in Sec. 5.1 has learned the universal pattern of jet radiation.

5.2.4 *Extrapolation with override*

In the previous section we have shown how truncating the final Bernoulli term from the likelihood loss allows the network to generate high-quality

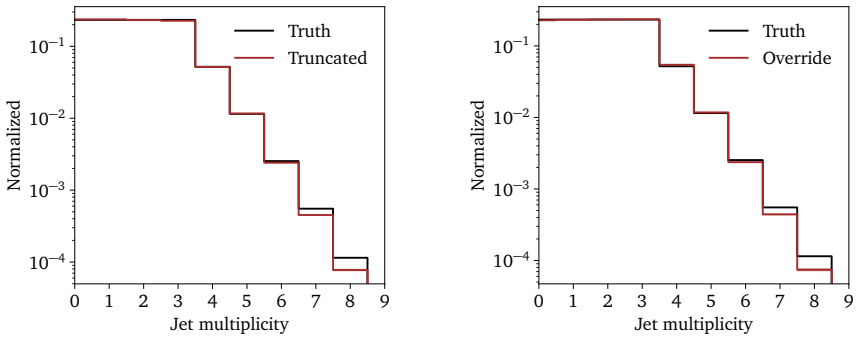

Figure 5.8: Jet multiplicity distributions learned using $\mathcal{L}_{\text{trunc}}$ (left) and $\mathcal{L}_{\text{override}}$ (right) trained on events with up to 6 jets.

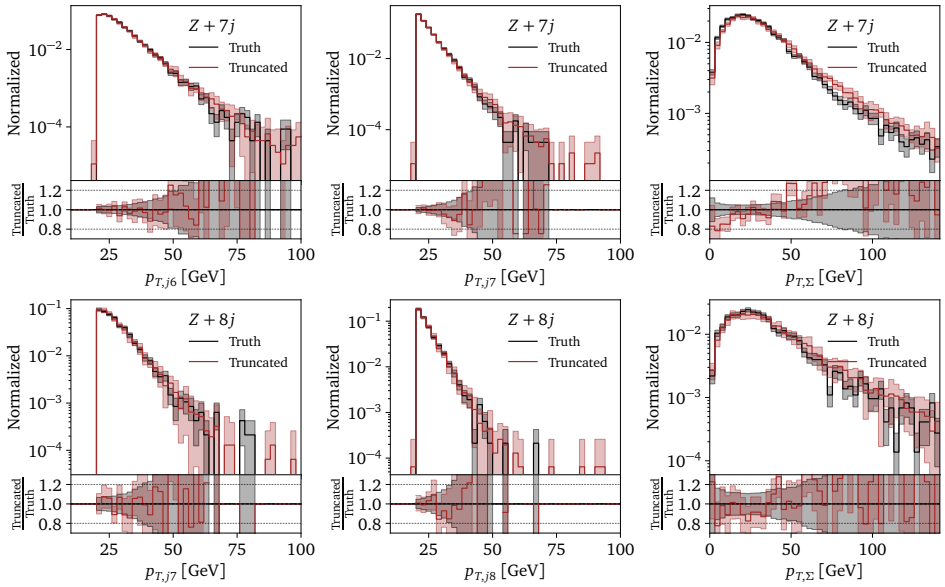

Figure 5.9: Selection of features $Z + 7$ and 8-jet events, trained with the truncated loss.

7-jet and 8-jet events. However, the extrapolation can be mis-calibrated. Because the transformer splitting predictions ρ_i are trained with a binary cross entropy, the optimal solution in terms of the non-splitting probability is the posterior

$$1 - \rho_i \approx p(\text{stop at } i | x_{1:i}) = \frac{p(x_{1:i}|\text{stop at } i)}{\sum_{n \geqslant i} p(x_{1:i}|\text{stop at } n)} \tag{5.26}$$

assuming a uniform prior for simplicity. When training on a dataset with maximum multiplicity n_{max}, the estimate is biased since the sum over n is missing terms above n_{max}. The same effect causes the transformer to stick to a constant splitting probability $\rho_{n_{\text{max}}} = 0$.

In an alternative approach we show that transverse momentum conservation can be used as an extra handle on the posterior. A violation of transverse momentum conservation can be induced by removing particles beyond the hard process and first k jets. The spread in center of momentum scales with k, so we can use transverse momentum conservation to statistically separate complete and incomplete events. Secondly, we note that the $p_{x,\Sigma}$ and $p_{y,\Sigma}$ distributions are roughly Gaussians with zero mean, and hence fully specified by their standard deviation.

In Fig. 5.10, we show the widths of the $p_{x,\Sigma}$ distributions as a function of the jet number, for complete events and for the hard process plus k jets. The widths obey an approximately linear scaling when considering a fixed number of jets, for complete events or otherwise. We can perform a linear fit to estimate the standard deviations for higher-multiplicity events, giving analytic expressions for the likelihoods in Eq.(5.26). We arrive at

$$\sigma(n; k) = (n - k)\, m_k + \sigma(k; k)\ ,$$

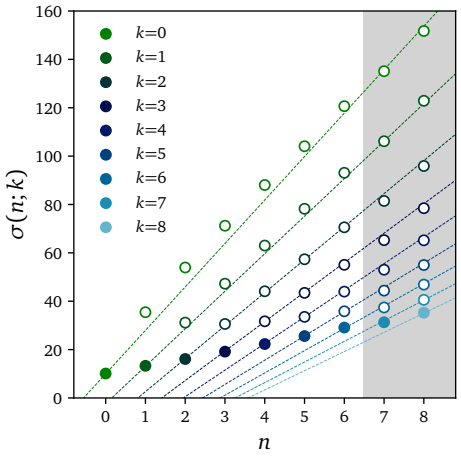

Figure 5.10: Standard deviations of $p_{x,\Sigma}$ for the muons and first k jets in $Z+n$-jet events. Filled circles indicate complete events, with $k = n$, while empty circles are incomplete. Points in the gray region are not used in any fit, but show the agreement of the extrapolation.

$$\begin{aligned} &\text{with} \quad \sigma(k;k) = 3.14k + 9.97\,, \\ &\text{and} \quad 1/m_k = 0.0088k + 0.056\,. \end{aligned} \tag{5.27}$$

The widths of completed events, $\sigma(k;k)$, are fit from the bottom row of filled circles in Fig. 5.10 up to $n = 6$. The gradients m_k of lines with constant k are fit using events up to $n = 5$. The fits are shown as dotted lines in Fig. 5.10, and we see that they extrapolate well to all partial k values in 7-jet and 8-jet events. Due to the rotation symmetry around the beam axis, the same values hold for $p_{y,\Sigma}$ and we can assume that the joint likelihoods are the product of 1D Gaussians.

$$\begin{aligned} p_{\text{fit}}(x_{1:k}|\text{stop at } n) = \mathcal{N}\left(p_{x,\Sigma}(x_{1:k}) \,\middle|\, 0, \sigma(n;k)\right) \\ \times \mathcal{N}\left(p_{y,\Sigma}(x_{1:k}) \,\middle|\, 0, \sigma(n;k)\right) \end{aligned} \tag{5.28}$$

Using Eq.(5.28) we can calculate target posteriors for an arbitrary maximum number of jets. This allows us to modify the likelihood loss by generalizing the Bernoulli splitting variable δ_{in} to a continuous variable $y_i \in [0,1]$ and override the troublesome $p_{\text{split}} = 0$ label for particle $n_{\max}$ by this estimate weighted by a hyperparameter λ,

$$\mathcal{L}_{\text{override}} = \left\langle -\sum_{i=1}^{n} \log p_{\text{kin}}(x_i|v_{i-1}) - \sum_{i=0}^{n} \lambda_i \log p_{\text{bin}}\left(y_i;\rho_i\right) \right\rangle_{x\sim p_{\text{data}}} \tag{5.29}$$

$$\text{with} \quad y_i(x_{1:n}) = \begin{cases} (1-\delta_{in}) & i < n_{\max} \\ 1 - p_{\text{fit}}(\text{stop at } i|x_{1:i}) & i = n_{\max} \end{cases}, \tag{5.30}$$

$$\text{and} \quad \lambda_i = 1 - (1-\lambda)\delta_{in_{\max}}\,. \tag{5.31}$$

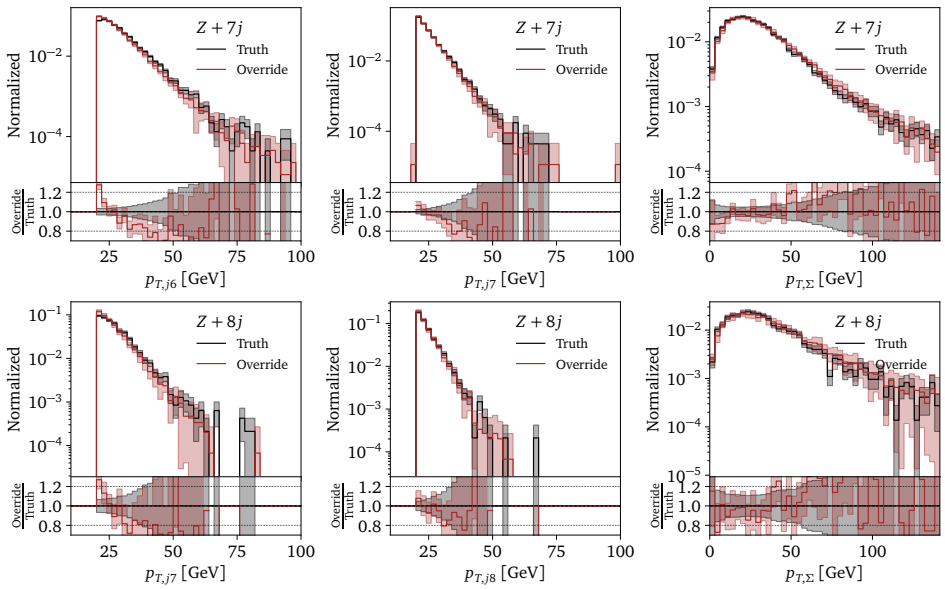

Figure 5.11: Selection of features Z + 7 and 8-jet events, trained with $\mathcal{L}_{\text{override}}$ on up to 6 jets.

The posterior $p_{\text{fit}}(\text{stop at } i|x_{1:i})$ is calculated using Eqs.(5.26) and (5.28) up to 8 jets. In practice, we find the best performance by including a staircase scaling prior when calculating the posterior. To match the dataset, we take $R_{(n+1)/n} = 0.225$ and cap the probabilities for $n < 3$. Note that the hyperparameter λ multiplies only the p_{bin} contribution for particle n_{max}.

We train a network with $\lambda = 0.2$ and show its event multiplicity distribution in Fig. 5.8 and network samples in Fig. 5.11. Similarly to the truncated loss, this override approach significantly increasing the fraction of higher-multiplicity events. looking at the kinematics, the p_T distributions now display an excess toward low values, but the global momentum correlation $p_{T,\Sigma}$ is reproduced to greater accuracy than previously. Once again, this demonstrates that autoregressive transformers can learn the universal nature of jet radiation.

5.3 OUTLOOK

The universality of splitting kernels and jet ratios in QCD provides the perfect physics question to see if appropriate generative networks can extrapolate. As an example, we study Z+jets events and the established staircase scaling of the jet number.

We employ an autoregressive transformer to learn a factorized likelihood for events across varying jet multiplicity. This autoregressive transformer sequentially predicts the kinematics of an additional jet along with the probability to radiate it. When trained the standard way, the transformer learns the kinematics of up to the 6 jets included in the training data with high fidelity. It also produces a small number of higher-multiplicity events with reasonable kinematics.

The first path towards extrapolation is to modify the training data with bootstrapping. This approach is straight-forward to adapt the multiplicity distribution, but some kinematic distributions are not learned very precisely.

Another way to extrapolate is to truncate the loss function and remove the explicit learning of the hard Sudakov factor for the highest multiplicity. Alternatively, we can override the hard Sudakov factor using physics information. We find that all both approaches are equally capable of generating high-quality events. These results establish that autoregressive transformers can learn the universal nature of jet radiation.

We emphasize that our study only shows that generative networks can extrapolate, given the right QCD properties of the training data. We expect the performance of all extrapolating networks to improve with technical advances.

6

A LORENTZ-EQUIVARIANT TRANSFORMER FOR ALL OF THE LHC

The research presented in this chapter is based on work in collaboration with Johann Brehmer, Víctor Bresó, Pim de Haan, Tilman Plehn, Huilin Qu and Jesse Thaler, and has been previously published in Refs. [95, 103]. All tables and figures as well as parts of the text are similar or identical to the content of these articles.

In this chapter, we introduce the Lorentz Geometric Algebra Transformer (L-GATr), a new general-purpose network architecture for high-energy physics. It is based on three design choices. First, L-GATr is equivariant with respect to the Lorentz symmetry.[1] It supports partial and approximate symmetries as found in some high-energy physics applications through symmetry-breaking inputs. Second, as representations, L-GATr uses the geometric (or Clifford) algebra over the four-vectors of special relativity. This algebra is based on the scalar and four-vector properties that LHC data are naturally parameterized in and extends them to higher orders, increasing the network capacity. Finally, L-GATr is a Transformer. It supports variable-length inputs, as found in many LHC problems, and even large models can be trained efficiently. Because it computes pairwise interactions through scaled dot-product attention, for which there are highly optimized backends like Flash Attention [59], the architecture scales particularly well to problems with many tokens or particles.

After reviewing the construction of L-GATr in Sec. 6.1, we demonstrate the versatility of L-GATr through three LHC case studies. In Sec. 6.2 we show how it allows us to efficiently learn precision surrogates for scattering amplitudes up to a $Z + 5$ gluon final state. Next, we show in Sec. 6.3 how we can improve transformer-based jet taggers with an equivariant setup. We show how L-GATr benefits from pre-training for the ultimate performance and can be generalized to multi-class tagging. Finally, in Sec. 6.4 we employ L-GATr inside a diffusion generator and show how it generates LHC events for instance for final states up to $t\bar{t} + 4$ jets better than all benchmarks. We provide a brief summary and outlook in Sec. 6.5.

6.1 LORENTZ-EQUIVARIANT GEOMETRIC ALGEBRA TRANSFORMER

In this section, we discuss the principles behind L-GATr and its most important features. L-GATr uses the geometric algebra, a mathematical framework that represents certain geometric objects and operations in a

1 One could extend L-GATr to the full Poincaré symmetry, which additionally includes space-time translations. However, this is not necessary for most particle-physics applications, as usually only the momentum, and not the absolute position, of particles is of interest. A key exception is the study of long-lived particles.

unified language. Using the language of geometric algebra, it is straightforward to build equivariant layers while retaining most of the structure from typical neural networks. In cases where the Lorentz symmetry is not completely realized, we show how reference inputs can be used to make L-GATr equivariant with respect to specific subgroups of the Lorentz group. Finally, we study how L-GATr scales with the phase space dimensionality, as compared to other network architectures.

6.1.1 *Spacetime Geometric Algebra*

A geometric algebra is defined as an extension of a vector space with an extra composition law — the geometric product [104]. The geometric product of two vectors x and y is decomposed into a symmetric and an antisymmetric contribution,

$$xy = \frac{\{x,y\}}{2} + \frac{[x,y]}{2} \,, \tag{6.1}$$

where the anti-commutator $\{x,y\}/2$ represents the usual inner product, and $[x,y]/2$ constitutes a new outer product. This second term defines the bivector, which can be understood geometrically as an area element of the plane spanned by x and y. The geometric product xy is an element of the algebra made up by the sum of a scalar and a bivector. Neither object is part of the original vector space, so the geometric product extends the original vector space. This extension can be carried out systematically by applying the geometric product repeatedly to the basis elements of the vector space.

To develop L-GATr, we focus on the spacetime algebra $\mathbb{G}_{1,3}$, built from the vector space $\mathbb{R}^4$ with the metric $g = \mathrm{diag}(1,-1,-1,-1)$. We choose as a basis for the vector space a set of four real vectors γ^μ, which satisfy the anti-commutation relation

$$\{\gamma^\mu, \gamma^\nu\} = 2g^{\mu\nu} \,. \tag{6.2}$$

This inner product establishes the basis elements as a set of orthogonal vectors and fixes their normalization. This happens to also be the defining property of the gamma matrices, the basis elements of the Dirac algebra used to describe spinor interactions. Both algebras are closely related, the only difference being that the spacetime algebra is defined over $\mathbb{R}^4$, whereas the Dirac algebra is defined over $\mathbb{C}^4$. This prescription fully recovers all the algebra properties presented in Ref. [95].

We now construct new elements of the algebra using the geometric product defined in Eq. (6.1). All higher-order elements can be characterized as antisymmetric products of γ^μ. We organize them in grades, defined by the number of γ^μ needed to express them. For instance, the antisymmetric tensor $\sigma^{\mu\nu}$ is generated from the geometric product of two γ^μ and consequently has grade two,

$$\gamma^\mu\gamma^\nu = \frac{\{\gamma^\mu,\gamma^\nu\}}{2} + \frac{[\gamma^\mu,\gamma^\nu]}{2} = g^{\mu\nu} + \sigma^{\mu\nu} \,. \tag{6.3}$$

Following Eq. (6.1), $\sigma^{\mu\nu}$ is a bivector, which can be interpreted as the plane opened by μ and ν in Minkowski space. We see that the symmetric term in the geometric product reduces the grade, while the antisymmetric term increases it. The whole product $\gamma^\mu\gamma^\nu$ is a sum of grade zero (scalar) and grade two. A generic element of the algebra that mixes grade information is called a multivector.

Moving on, the geometric product of three vectors $\gamma^\mu\gamma^\nu\gamma^\rho$ contains the antisymmetric tensor $\epsilon_{\mu\nu\rho\sigma}\gamma^\mu\gamma^\nu\gamma^\rho$ as a trivector or axial vector. The product of all four γ^μ leads us to the pseudoscalar

$$\gamma^5 = \gamma^0\gamma^1\gamma^2\gamma^3 \equiv \frac{1}{4!}\epsilon_{\mu\nu\rho\sigma}\gamma^\mu\gamma^\nu\gamma^\rho\gamma^\sigma. \tag{6.4}$$

Pseudoscalars act as parity reversal operations on any object and can be used to write axial vectors as $\gamma^\mu\gamma^5$. The missing factor i compared to the usual definition of γ^5 indicates the slight difference between the complex Dirac algebra and our real spacetime algebra.

Geometric products with more than four γ^μ can be reduced to lower-grade structures. Combining all these elements, we can express any multivector of the algebra as

$$x = x^S\,1 + x^V_\mu\,\gamma^\mu + x^B_{\mu\nu}\,\sigma^{\mu\nu} + x^A_\mu\,\gamma^\mu\gamma^5 + x^P\,\gamma^5 \quad \text{with} \quad \begin{pmatrix} x^S \\ x^V_\mu \\ x^B_{\mu\nu} \\ x^A_\mu \\ x^P \end{pmatrix} \in \mathbb{R}^{16}\,. \tag{6.5}$$

In this representation, we only include the nonzero and independent entries in the antisymmetric bivector. Multivectors can be used to represent both spacetime objects and Lorentz transformations. For instance, particles are characterized by their type (i.e. particle identification, or PID) and their 4-momentum p^μ,

$$x^S = \text{PID} \qquad x^V_\mu = p_\mu \qquad x^T_{\mu\nu} = x^A_\mu = x^P = 0\,. \tag{6.6}$$

Using this convenient representation, the spacetime algebra naturally structures relevant objects like parity-violating transition amplitudes. The matrix element $\mathcal{M}$ is a function of 4-momenta and can be decomposed into parity-even and parity-odd terms before it gets squared,

$$|\mathcal{M}|^2 = |\mathcal{M}_E|^2 + |\mathcal{M}_O|^2 + 2\,\mathrm{Re}\,(\mathcal{M}_E^*\mathcal{M}_O)\,. \tag{6.7}$$

The first two terms represent a scalar function of the 4-momenta, while the last is a pseudoscalar function. To calculate this amplitude in the spacetime algebra, we first embed each 4-momentum into a multivector $x^i = p^i_\mu\gamma^\mu$. Using these multivectors as inputs, the squared amplitude can be obtained through a sequence of algebra operations. The result of this calculation will also be a multivector, namely

$$|\mathcal{M}|^2 = x^S 1 + x^P \gamma^5 \qquad \text{with} \qquad \begin{aligned} x^S 1 &= |\mathcal{M}_E|^2 + |\mathcal{M}_O|^2 \\ x^P \gamma^5 &= 2\,\mathrm{Re}\,(\mathcal{M}_E^* \mathcal{M}_O) \ . \end{aligned} \tag{6.8}$$

The geometric algebra explicitly separates the scalar and pseudoscalar components of the squared amplitude, highlighting their respective geometric significance.

The geometric algebra also allows us to perform operations on spacetime objects. Lorentz transformations act as

$$\Lambda_v(x) = vxv^{-1} \ , \tag{6.9}$$

where v is a multivector representing an element of the Lorentz group acting on the algebra element x, and v^{-1} representings the corresponding inverse. The representation v of a Lorentz transformation is built by a simple rule: a multivector encoding an object that is invariant under a Lorentz transformation will also represent the transformation itself. This gives a dual interpretation to spacetime algebra elements as, both, geometric objects and Lorentz transformations.

For instance, boosts along the z-axis are generated by σ^{03}, which also represents a plane in time vs. z-direction. The multivector for such a boost with rapidity ω reads

$$v = e^{\omega\,\sigma^{03}} = 1\cosh\omega + \sigma^{03}\sinh\omega \ . \tag{6.10}$$

If we apply this boost to a particle moving in z-direction, $x = E\gamma^0 + p_z\gamma^3$, the transformation in Eq. (6.9) gives us

$$vxv^{-1} = (E\cosh\omega - p_z\sinh\omega)\gamma^0 + (p_z\cosh\omega - E\sinh\omega)\gamma^3 \ . \tag{6.11}$$

This is exactly what we expect from the Lorentz boost. The algebra representation allows us to apply this boost on any object in the geometric algebra, not just vectors. From Eq. (6.9) and the properties of the geometric product, we see that Lorentz transformations will never mix grades. Each algebra grade transforms under a separate sub-representation of the Lorentz group.

The main limitation of the geometric algebra approach is that the spacetime algebra $G_{1,3}$ covers only a limited range of Lorentz tensor representations. For instance, this formalism can not represent symmetric rank-2 tensors. For most LHC applications, though, one does not encounter higher-order tensor representations as inputs or outputs, so this is not a substantial limitation. Whether higher-order tensors might be needed for internal representations within a network is an open question [105].

6.1.2 *Constructing a Lorentz-Equivariant Architecture*

Based on the multivector representation, we now construct the corresponding transformer network L-GATr. It is equivariant under Lorentz group transformations Λ

$$\text{L-GATr}\Big(\Lambda(x)\Big) = \Lambda\Big(\text{L-GATr}(x)\Big) \,. \tag{6.12}$$

We take advantage of the fact that multivector grades form sub-representations of the Lorentz group, i. e.all multivector components of the same grade transform equally under all network operations, whereas different grades transform differently [95, 106].

The L-GATr architecture uses variations of the standard transformer operations Linear, Attention, LayerNorm, and Activation, adapted to process multivectors [58, 107]. As usual for transformers, the input x and output $\text{L-GATr}(x)$ are unordered sets of n_t ordered lists of n_c multivector channels

$$x_{ic} = \begin{pmatrix} x^S_{ic} \\ x^V_{\mu,ic} \\ x^B_{\mu\nu,ic} \\ x^A_{\mu,ic} \\ x^P_{ic} \end{pmatrix} \qquad i = 1,...,n_t \quad c = 1,...,n_c \,. \tag{6.13}$$

We call the set elements $x_i = \{x_{ic} : c = 1,...,n_c\}$ tokens, where each token can represent a particle. In the network, every operation will have multivectors as inputs and outputs. The full L-GATr architecture is built as

$$\begin{aligned} \bar{x} &= \text{LayerNorm}(x) \\ \text{AttentionBlock}(x) &= \text{Linear} \circ \text{Attention}(\text{Linear}(\bar{x}), \text{Linear}(\bar{x}), \text{Linear}(\bar{x})) + x \\ \text{MLPBlock}(x) &= \text{Linear} \circ \text{Activation} \circ \text{Linear} \circ \text{GP}(\text{Linear}(\bar{x}), \text{Linear}(\bar{x})) + x \\ \text{Block}(x) &= \text{MLPBlock} \circ \text{AttentionBlock}(x) \\ \text{L-GATr}(x) &= \text{Linear} \circ \text{Block} \circ \text{Block} \circ \cdots \circ \text{Block} \circ \text{Linear}(x) \,. \end{aligned} \tag{6.14}$$

We define the modified transformer operations in some detail:

- For the linear layers, we use the fact that equivariant operations on multivectors process components within the same grade equally. We use the projection $\langle\cdot\rangle_k$ to extract the k-grade and apply different learnable coefficients for each grade. As a result, the most general linear combination of independently-transforming multivector components is

$$\text{Linear}(x) = \sum_{k=0}^{4} \nu_k \langle x\rangle_k \quad \left(+ \sum_{k=0}^{4} w_k \gamma^5 \langle x\rangle_k \right) \,, \tag{6.15}$$

  where $\nu, w \in \mathbb{R}^5$ are learnable parameters and k runs over the five algebra grades. The second term is optional and breaks the symmetry down to the special orthochronous Lorentz group, the fully-connected subgroup that leaves out parity and time reversal. In this subgroup,

discrete transformations are not present, so any pair of algebra elements that differ by a γ^5 factor can be linearly mixed without breaking equivariance.

- We extend scaled dot-product attention such that it can be applied to multivectors

$$\text{Attention}(q, k, v)_{ic} = \sum_{j=1}^{n_t} \text{Softmax}_j \left(\sum_{c'=1}^{n_c} \frac{\langle q_{ic'}, k_{jc'} \rangle}{\sqrt{16 n_c}} \right) v_{jc} \,, \tag{6.16}$$

  where n_c is the number of multivector channels and $\langle \cdot, \cdot \rangle$ is the $G_{1,3}$ inner product. This inner product can be pre-computed as a list of signs and a Euclidean inner product, allowing us to use standard transformer implementations.

- Layer normalization on multivectors is non-trivial because the $G_{1,3}$ norm can have zero and negative contributions. For this reason, we define layer normalization using the absolute value of the inner product for each grade separately

$$\text{LayerNorm}(x) = \frac{x}{\sqrt{\frac{1}{n_c} \sum_{c=1}^{n_c} \sum_{k=0}^{4} \left| \left\langle \langle x_c \rangle_k, \langle x_c \rangle_k \right\rangle \right| + \epsilon}} \,, \tag{6.17}$$

  where ϵ is a normalization constant and n_c is the number of multivector channels.

- Activation functions applied directly on the multivectors break the equivariance. We employ scalar-gated activation functions [106], where the nonlinearity only acts on the scalar component of the multivector $\langle x \rangle_0$. Specifically, we use the scalar-gated GELU [108] activation function

$$\text{Activation}(x) = \text{GELU}(\langle x \rangle_0)\, x \,. \tag{6.18}$$

- Finally, the geometric algebra allows for another source of nonlinearity, the geometric product

$$\text{GP}(x, y) = xy \qquad \text{with} \qquad \text{GP}(vxv^{-1}, vyv^{-1}) = v\,\text{GP}(x, y)\,v^{-1} \,, \tag{6.19}$$

  which is equivariant itself.

These operations strictly generalize standard scalar transformers to the multivector representation, as illustrated in Table 6.1. We supplement the list of multivector channels with extra scalar channels to allow a smooth transition to standard transformers that solely rely on scalar channels. Moreover, it provides a handle to feed large amounts of scalar information to the network without overloading the multivector channels.

Layer type	Transformer	L-GATr
Linear(x)	$vx + w$	$\sum_{k=0}^{4} v_k \langle x \rangle_k$
Attention$(q, k, v)_{ic}$	$\sum_{j=1}^{n_t} \text{Softmax}_j \left(\sum_{c'=1}^{n_c} \frac{q_{ic'} k_{jc'}}{\sqrt{n_c}} \right) v_{jc}$	$\sum_{j=1}^{n_t} \text{Softmax}_j \left(\sum_{c'=1}^{n_c} \frac{\langle q_{ic'}, k_{jc'} \rangle}{\sqrt{16 n_c}} \right) v_{jc}$
LayerNorm(x)	$x \left[\frac{1}{n_c} \sum_{c=1}^{n_c} x_c^2 + \epsilon \right]^{-1/2}$	$x \left[\frac{1}{n_c} \sum_{c=1}^{n_c} \sum_{k=0}^{4} \left\lvert \left\langle \langle x_c \rangle_k, \langle x_c \rangle_k \right\rangle \right\rvert + \epsilon \right]^{-1/2}$
Activation(x)	GELU(x)	GELU$(\langle x \rangle_0) x$
GP(x, y)	–	xy

Table 6.1: Comparison of transformer layers and L-GATr layers. The arguments x, y, q, k, v are scalars for the transformer, and multivectors for L-GATr. The second term in the L-GATr linear layer is optional and breaks the Lorentz group down to its fully connected subgroup.

6.1.3 *Breaking Lorentz Symmetry*

In many LHC contexts, Lorentz symmetry is only partially preserved. L-GATr can apply partial symmetry breaking in a tunable manner by including reference multivectors as additional inputs. Any network operation that involves such a reference vector will violate equivariance, breaking the symmetry group to a subgroup where the reference direction is fixed. This defines a partial symmetry breaking without altering the structure of the network. The network always has the option to tune out the reference vectors when they are not needed. Reference vectors are appended for each L-GATr input x in the same way, either as extra tokens, or as extra channels within each token.

For instance, the LHC beam direction breaks the Lorentz group to the subgroup of rotations around and boosts along the beam axis [109–112]. The natural reference vector is this beam direction, which can be either implemented as two vectors $x_{\pm}^{V} = (0, 0, 0, \pm 1)$, or one bivector representing the $x-y$ plane, $x_{12}^{B} = 1$. We find similar performance for both choices. Generally, we can break the Lorentz group to the subgroup of rotations in three-dimensional space $SO(3)$ using the reference multivector $x^{V} = (1, 0, 0, 0)$.

We include such reference multivectors as extra tokens for jet tagging in Sec. 6.3, and as extra channels for generation in Sec 6.4. In both cases, this symmetry breaking is crucial, and the specific way it is implemented has a strong impact on the network performance.

We find it beneficial to add more ways of breaking the symmetry that are formally equivalent to the reference multivectors discussed above. For jet tagging, we include additional kinematic inputs like $p_T, E, \Delta R$ embedded as scalars. These variables are only invariant under the subgroup of rotations around the beam axis, and L-GATr can reconstruct them based on the particles and reference multivectors. For event generation, we extract the m and p_T CFM-velocity components from scalar output channels of L-GATr and use them to overwrite the equivariantly predicted velocity components. We explain these aspects further in Sec. 6.3, 6.4.

6.1.4 *Scaling with the Number of Particles*

Fully connected graph neural networks bear a very close resemblance to transformers [113]; both process data as sets of tokens, both respect full or partial permutation symmetry, and both can be turned equivariant [95, 111, 112, 114]. Resource efficiency is where the two architectures differ most. To quantify it, we measure the scaling of speed and memory consumption of a standard transformer, L-GATr, and the Clifford Group Equivariant Neural Network (CGENN) [111], which is a graph network built on geometric algebra representations. We expect the CGENN to represent the strengths and limitations of equivariant graph networks. We execute network forward passes with synthetic data on an H100 GPU. Each measurement is done with a single event consisting of a varying number of particles. To ensure fairness, all networks consist of a single network block, for the transformers the attention receives inputs with 72 channels from 4 attention heads, and the CGENN is set up to be as close as possible to L-GATr. Namely, both contain 8 multivector channels and 16 scalar channels.

In the left panel of Fig. 6.1, we see that the evaluation time of all networks is independent of the number of tokens in the few-token regime, but it eventually scales quadratically. This transition happens when attention or message passing, rather than other parallelizable operations, becomes the limiting factor. L-GATr scales like a standard transformer in the many-token regime because they both use the same attention module. For few tokens, L-GATr is slower because of the more expensive linear layers. The CGENN is slower than L-GATr for few tokens, and the quadratic scaling due to the expensive message passing operation takes off sooner.

As can be seen in the right panel of Fig. 6.1, for many particles L-GATr and the standard transformer display the same linear scaling in memory usage with the number of tokens since they use the same attention module.

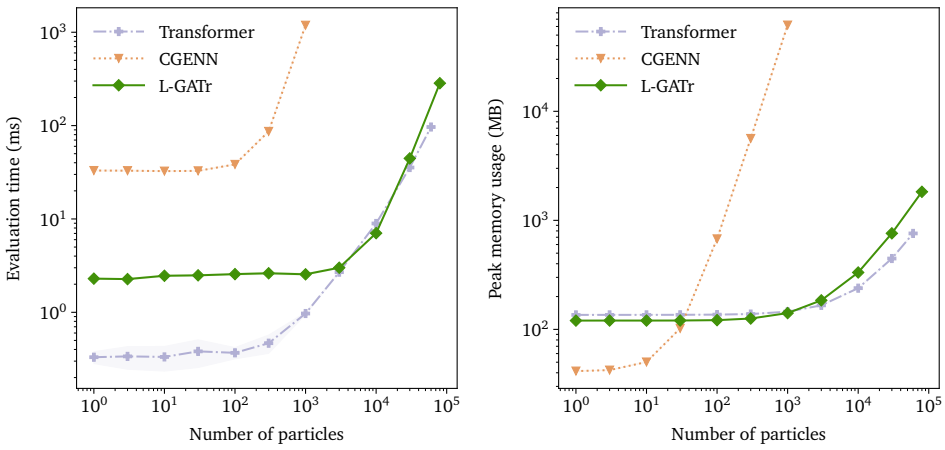

Figure 6.1: Scaling behavior of L-GATr, a standard transformer, and the equivariant graph network CGENN. The left panel was already discussed in Ref. [95].

In contrast, the CGENN scales quadratically in this regime and runs out of memory already for 1000 particles. We attribute this to the different degree of optimization in the architectures. For L-GATr we use FlashAttention [59], heavily optimized for speed and memory efficiency. Graph neural networks are often optimized for sparsely connected graphs, so the efficiency of the standard implementation degrades for fully connected graphs.

6.2 L-GATR FOR AMPLITUDE REGRESSION

Our first L-GATr case study is for amplitude regression. Partonic scattering amplitudes can be calculated exactly as a function of phase space. However, their evaluation can become very expensive if we include high-order corrections and a large number of external particles. Amplitude surrogates as part of standard event generators speed up these precision predictions [39, 40, 115, 116]. However, standard neural networks struggle to reach sufficient accuracy for realistic numbers of external particles. L-GATr uses the partial permutation symmetry of particles in the process to efficiently scale to high multiplicities, and it guarantees the Lorentz invariance of the amplitude. Extending the studies performed in Ref. [95], we demonstrate its utility for the partonic processes

$$q\bar{q} \to Z + n\,g, \qquad n = 1 \ldots 4\,. \tag{6.20}$$

We train L-GATr networks with a standard MSE loss to predict the corresponding squared amplitudes A from the initial and final state 4-momenta.

We generate 4×10^5 training data points for each multiplicity up to 4 gluons with MadGraph [117]. First, we use a standard run to generate unweighted phase space points; second, we apply the standalone module

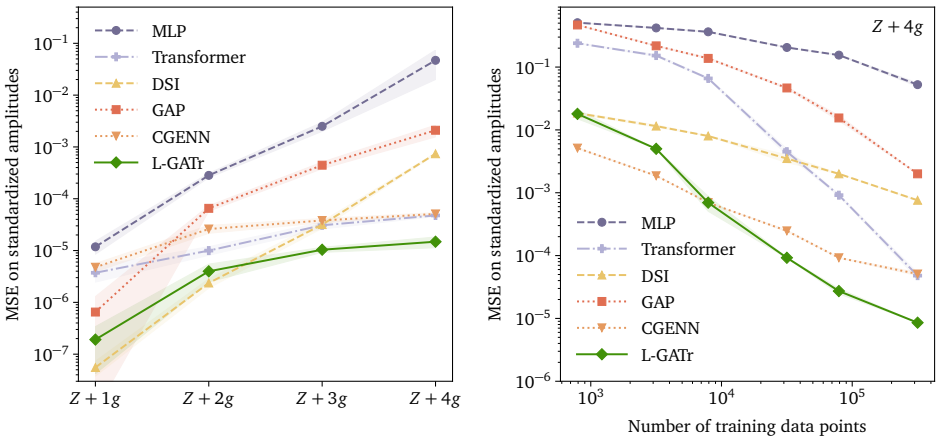

Figure 6.2: Left: prediction error from L-GATr and all baselines for $Z + ng$ amplitudes with increasing particle multiplicity. All networks are trained on 4×10^5 points. Right: prediction error as a function of the training dataset size. Error bands are based on the mean and standard deviation of five random seeds. These figures are also included in Ref. [95].

to compute the squared amplitude values. To avoid divergences, we require globally

$$p_T > 20 \text{ GeV} \qquad \text{and} \qquad \Delta R > 0.4 \tag{6.21}$$

for all final-state objects. We train on standardized logarithmic amplitudes

$$\mathcal{A} = \frac{\log A - \overline{\log A}}{\sigma_{\log A}} \,. \tag{6.22}$$

In addition to the L-GATr surrogate we also train a comprehensive set of benchmarks:

- a standard MLP;
- a standard transformer [58], as L-GATr without equivariance;
- the Geometric Algebra Perceptron (GAP), as L-GATr without transformer structure;
- a deep sets network (DSI) [118] combining partial Lorentz and permutation equivariance;
- the CGENN [111] as an equivariant graph network operating on multivectors like L-GATr.

Details about the implementation and training can be found in the Appendix A.3.

Performance

Using the MSE loss as a quality metric, we compare the performance of L-GATr to the baselines in the left panel of Fig. 6.2. Overall, transformer and graph networks scale better with the number of external particles. We find that L-GATr is roughly on par with the leading DSI network for a small number of gluons, but its improved scaling gives it the lead for higher-multiplicity final states. We find similar performance for L-GATr when we train a single network on all processes jointly.

Next, we show in the right panel of Fig. 6.2 how the performance scales with the size of the training dataset. L-GATr stands as a top performer on all training regimes. In particular for small training sets, L-GATr and CGENN are very efficient thanks to their equivariant operations.

6.3 L-GATR FOR JET TAGGING

Jet tagging is, arguably, the LHC task which is currently impacted most by modern ML. Two approaches stand out as top performers: transformer-based architectures and equivariant networks. In this section, we show how L-GATr sets a new record for jet tagging by combining the merits of both ideas. All results related to pre-training and multiclass tagging are new to this paper.

Top tagging

We first study the performance of L-GATr on the top tagging challenge [28], a representative and extensively studied jet tagging task at the LHC. The results in this section were first presented in Ref. [95].

The top tagging dataset, originally produced for Ref. [119], consists of 2 M top quark and QCD jets with

$$p_{T,j} = 550 \dots 650 \text{ GeV} , \tag{6.23}$$

generated with Pythia 8 [80] and interfaced with Delphes for detector simulation [120] using the default ATLAS card at that time. We train and evaluate the L-GATr tagger on this dataset following the standard train/validation/test splitting of 1.2/0.4/0.4 M. Details about the network implementation and the training method are provided in Appendix A.3.

We compare our L-GATr tagger with the following baselines:

- LorentzNet [112], an equivariant graph network based on functions of the momentum invariants as coefficients for 4-momenta inputs;
- PELICAN [110], an alternative equivariant graph network based on momentum invariants and permutation equivariant aggregation functions;
- CGENN [111], an equivariant graph network operating on multivectors;
- ParT [29], a transformer that includes pairwise interaction features as an attention bias; and

Network	Accuracy	AUC	$1/\epsilon_B$ ($\epsilon_S = 0.5$)	$1/\epsilon_B$ ($\epsilon_S = 0.3$)
TopoDNN [121]	0.916	0.972	–	295 ± 5
LoLa [122]	0.929	0.980	–	722 ± 17
N-subjettiness [123]	0.929	0.981	–	867 ± 15
PFN [124]	0.932	0.9819	247 ± 3	888 ± 17
TreeNiN [125]	0.933	0.982	–	1025 ± 11
ParticleNet [126]	0.940	0.9858	397 ± 7	1615 ± 93
ParT [29]	0.940	0.9858	413 ± 16	1602 ± 81
MIParT [127]	0.942	0.9868	505 ± 8	2010 ± 97
LorentzNet* [112]	0.942	0.9868	498 ± 18	2195 ± 173
CGENN* [111]	0.942	0.9869	500	2172
PELICAN* [110]	0.9426 ± 0.0002	0.9870 ± 0.0001	–	2250 ± 75
L-GATr* [95]	0.9423 ± 0.0002	0.9870 ± 0.0001	540 ± 20	2240 ± 70
ParticleNet-f.t. [127]	0.942	0.9866	487 ± 9	1771 ± 80
ParT-f.t. [127]	0.944	0.9877	691 ± 15	2766 ± 130
MIParT-f.t. [127]	0.944	0.9878	640 ± 10	2789 ± 133
L-GATr-f.t.* (new)	0.9446 ± 0.0002	0.98793 ± 0.00001	651 ± 11	2894 ± 84

Table 6.2: Top tagging accuracy, AUC, and background rejection $1/\epsilon_B$ for the standard dataset [28, 119]. Lorentz-equivariant methods are indicated with an asterisk, and fine-tuning methods are separated with a horizontal line. Our error bars are based on the mean and standard deviation of five random seeds.

Beam	Time	Embedding	AUC	$1/\epsilon_B$ ($\epsilon_S = 0.3$)
–	–	Token	0.9844	1422
$x_3^V = \pm 1$	–	Token	0.9850	1905
–	$x_0^V = 1$	Token	0.9865	1923
$x_{12}^B = x_{13}^B = x_{23}^B = 1$	$x_0^V = 1$	Token	0.9863	2009
$x_{12}^B = 1$	$x_0^V = 1$	Channel	0.9865	2060
$x_0^V = 1, x_3^V = \pm 1$	$x_0^V = 1$	Token	0.9869	2114
$x_3^V = \pm 1$	$x_0^V = 1$	Token	0.9869	2152
$x_{12}^B = 1$	$x_0^V = 1$	Token	0.9870	2240

Table 6.3: Symmetry breaking. We compare the L-GATr performance on the top tagging dataset from Ref. [128] using different Lorentz symmetry breaking schemes. The last line is our default used in all other tagging experiments.

- MIParT [127], an extension of ParT with specialized blocks that focus only on interaction features.

In Tab. 6.2, we see how L-GATr is at least on par with the leading equivariant baselines, as shown already in Ref. [95].

A key ingredient for the optimization of L-GATr is the symmetry breaking prescription. For all our tests, we include two reference vectors as extra tokens: the beam direction as the $x-y$ plane bivector, $x_{12}^B = 1$, and the time reference $x^V = (1, 0, 0, 0)$, which gives the network a handle to break the symmetry down to $SO(3)$. We provide a comparison between multiple reference vector options in Tab. 6.3, where we test their impact on a top tagging network. From it, it is clear that both the beam direction and the time reference significantly contribute to boosting the tagging performance.

Multi-class tagging on JetClass

We further study L-GATr for multiclass tagging with the JetClass dataset [29]. JetClass covers a wide variety of jet signatures. Its signal events consist of jets arising from multiple decay modes of top quarks, W, Z and Higgs bosons; its background events are made up of light quark and gluon jets. All types of events are generated with MadGraph [10] and Pythia [80], and detector effects are simulated with Delphes [129] using the default CMS card. A kinematic cut

$$p_{T,j} = 500...1000 \text{ GeV} \qquad \text{and} \qquad |\eta_j| < 2.0 \tag{6.24}$$

is applied to all jets in the dataset. In total, JetClass contains 100 M jets equally distributed across 10 classes.

JetClass provides input features of four main categories: the 4-momenta of the jet particles, kinematic variables like ΔR and $\log p_T$ that can be derived from the 4-momenta, particle identification variables, and trajectory displacement variables. When passing them through L-GATr, all features besides the 4-momenta are embedded to the network as scalar channels.

	All classes		$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to 4q$	$H \to l\nu q\bar{q}'$	$t \to bq\bar{q}'$	$t \to bl\nu$	$W \to q\bar{q}'$	$Z \to q\bar{q}$
	Accuracy	AUC	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99.5\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$
ParticleNet [126]	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT [29]	0.861	0.9877	10638	4149	123	1864	5479	32787	15873	543	402
MIParT [127]	0.861	0.9878	10753	4202	123	1927	5450	31250	16807	542	402
L-GATr	0.866	0.9885	12987	4819	128	2311	6116	47619	20408	588	432

Table 6.4: Tagging accuracy, AUC, and background rejection $1/\epsilon_B$ for the JetClass dataset [29]. The AUC is computed as the average of all possible pairwise combinations of classes, and the acceptances are computed by comparing each signal against the background class.

	All classes		$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to 4q$	$H \to l\nu q\bar{q}'$	$t \to bq\bar{q}'$	$t \to bl\nu$	$W \to q\bar{q}'$	$Z \to q\bar{q}$
	Accuracy	AUC	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99.5\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$
ParticleNet (2 M)	0.828	0.9820	5540	1681	90	662	1654	4049	4673	260	215
ParticleNet (10 M)	0.837	0.9837	5848	2070	96	770	2350	5495	6803	307	253
ParticleNet (100 M)	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT (2 M)	0.836	0.9834	5587	1982	93	761	1609	6061	4474	307	236
ParT (10 M)	0.850	0.9860	8734	3040	110	1274	3257	12579	8969	431	324
ParT (100 M)	0.861	0.9877	10638	4149	123	1864	5479	32787	15873	543	402
MIParT (2 M)	0.837	0.9836	5495	1940	95	819	1778	6192	4515	311	242
MIParT (10 M)	0.850	0.9861	8000	3003	112	1281	3650	16529	9852	440	336
MIParT (100 M)	0.861	0.9878	10753	4202	123	1927	5450	31250	16807	542	402
L-GATr (2 M)	0.839	0.9842	6623	2294	99	981	1980	8097	4902	346	276
L-GATr (10 M)	0.859	0.9875	9804	3883	120	1791	4255	24691	13333	506	373
L-GATr (100 M)	0.866	0.9885	12987	4819	128	2311	6116	47619	20408	588	432

Table 6.5: Tagging accuracy, AUC, and background rejection $1/\epsilon_B$ on different sizes of the JetClass dataset [29]. Metrics from other models are taken from their published results [29, 127].

We use the same architecture as we did for top tagging in all our tests (see Appendix A.3).

We present the results in Tab. 6.4, including reference metrics from Refs. [29, 127]. The L-GATr tagger achieves a significant improvement over the previous state-of-the-art, ParT and MIParT, in essentially all signal types. In a separate study, we have checked that the quality of L-GATr predictions steadily increases as we add more features to the training data. We also train L-GATr with subsets of the dataset to test its data efficiency. As we can see in the left panel in Fig. 6.3 and Table 6.5, L-GATr achieves a performance similar to the non-equivariant ParT and MIParT taggers even if trained with only 10% of all available jets.

Top tagging with JetClass pre-training

The large capacity of transformer architectures motivates using pre-training to further improve the tagging performance [29]. To this end, we pre-train L-GATr on the JetClass dataset and fine-tune it for top tagging. The pre-training uses the same setup as the JetClass training, but the inputs are limited to the 4-momenta and the derived kinematic variables, as those are the only available features in the top tagging dataset.

We follow Ref. [29] for the pre-training and fine-tuning procedures. Once the network is pre-trained on the large dataset, we fine-tune it by switching the last layer of the network to map to a single output channel and re-initialize its weights. During fine-tuning, the pre-trained model weights have to be updated with a smaller learning rate than the new ones, otherwise the network might dismiss all information from the pre-training. Further details about the fine-tuning setup are discussed in Appendix A.3.

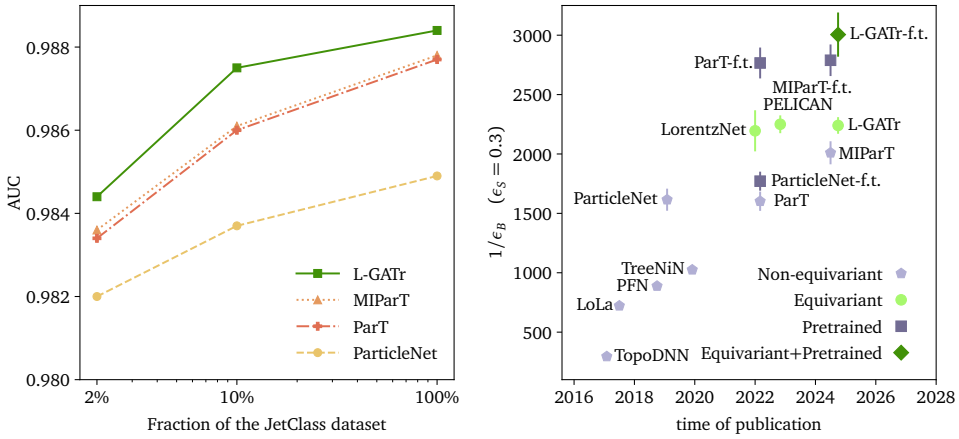

Figure 6.3: AUC metric on JetClass as a function of the training dataset fraction (left) and the history of top taggers (right).

We show the results from fine-tuned L-GATr in Tab. 6.2, where we compare different fine-tuned baselines. L-GATr matches the performance of the best fine-tuned networks in the literature across all metrics.[2] To further illustrate the impact of combining equivariance and pre-training, we summarize the historical progress in top tagging in the right panel of Fig. 6.3.

6.4 L-GATR FOR EVENT GENERATION

Generating LHC events is a key benchmark for neural network architectures, required for end-to-end generation, neural importance sampling, generative unfolding [32, 33, 131–134], and optimal inference [34, 35]. For all these tasks we should reach per-mille-level, or at the very least percent-level accuracy on the underlying phase space density. We use the L-GATr architecture to take advantage of the approximate symmetries and to improve the scaling for increasing numbers of final state particles. Our reference process is

$$pp \to t_h \bar{t}_h + n\,j, \qquad n = 0 \ldots 4\,, \tag{6.25}$$

with both top quarks decaying hadronically. It is simulated with MadGraph3.5.1, consisting of MadEvent [135] for the underlying hard process, Pythia8 [80] for the parton shower, Delphes3 [120] for the detector simulation, and the anti-k_T jet reconstruction algorithm [21] with $R = 0.4$ as implemented in FastJet [136]. We use Pythia without multi-parton interactions and the default ATLAS detector card. We apply the phase space cuts

$$p_{T,j} > 22\ \text{GeV} \qquad \Delta R_{jj} > 0.5 \qquad |\eta_j| < 5\,, \tag{6.26}$$

2 In this table, we see that the prediction made in the title of Ref. [130] turned out accurate.

and require two b-tagged jets. The events are reconstructed with a χ^2-based algorithm [137], and identical particles are ordered by p_T. The sizes of the $t\bar{t} + n\,j$ datasets reflect the frequency of the respective processes, resulting in 9.8M, 7.2M, 3.7M, 1.5M and 480k events for $n = 0 \ldots 4$. We train separate networks for each multiplicity, to allow for a direct comparison between different architectures, but we emphasize that transformers can also be trained jointly on all multiplicities [8]. This is particularly useful in the case of limited training data, because transformers can transfer information across multiplicities. The results presented here were briefly discussed in Ref. [95], but without proper benchmarking.

Conditional flow matching (CFM)

Continuous normalizing flows [138] learn a continuous transition $x(t)$ between a simple latent distribution $x_1 \sim p_{\text{latent}}(x_1)$ and a phase space distribution $x_0 \sim p_{\text{data}}(x_0)$. Mathematically, they build on two equivalent ways of describing a diffusion process, using either an ODE or a continuity equation [2, 8]

$$\frac{dx(t)}{dt} = v(x(t), t) \quad \text{or} \quad \frac{\partial p(x,t)}{\partial t} = -\nabla_x \left[v(x(t), t) p(x(t), t) \right] , \tag{6.27}$$

with the same CFM-velocity field $v(x(t), t)$. The diffusion process $t = 0 \to 1$ interpolates between the phase space distribution $p_{\text{data}}(x_0)$ and the base distribution $p_{\text{latent}}(x_1)$,

$$p(x,t) \to \begin{cases} p_{\text{data}} x_0) & t \to 0 \\ p_{\text{latent}}(x_1) & t \to 1 \,. \end{cases} \tag{6.28}$$

To train the continuous normalizing flow with conditional flow matching [139, 140], we employ a simple linear interpolation

$$x(t) = (1-t)x_0 + t x_1 \to \begin{cases} x_0 & t \to 0 \\ x_1 & t \to 1 \,. \end{cases} \tag{6.29}$$

and train the network with parameters θ on a standard MSE loss to encode the (CFM-)velocity

$$v_\theta \left((1-t)x_0 + t x_1, t \right) \approx x_1 - x_0 . \tag{6.30}$$

The network has to learn to match the un-conditional velocity field on the left hand side to the conditional velocity field on the right hand side. We then generate phase space configurations using a fast ODE solver via

$$x_0 = x_1 - \int_0^1 dt \; v_\theta(x(t), t). \tag{6.31}$$

L-GATr velocity

We compare the L-GATr performance with a set of leading benchmark architectures, all based on a CFM generator, with

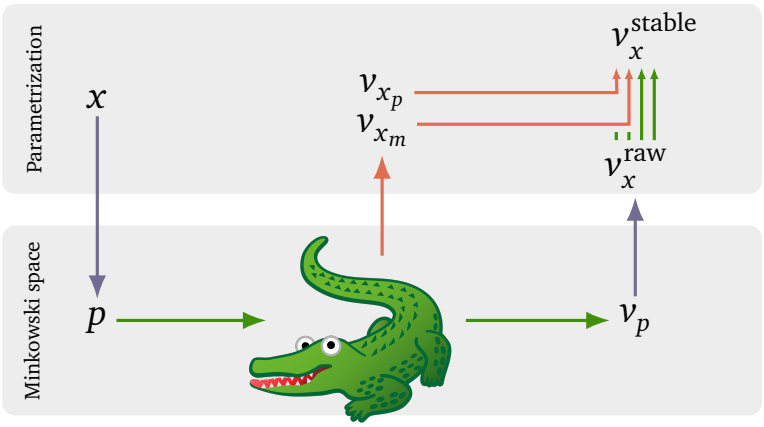

Figure 6.4: To construct the L-GATr velocity, we extract equivariantly predicted multivectors and symmetry-breaking scalars. We go back and forth between the parametrization x and Minkowski space p using the mapping f from Eq. (6.32).

- a standard MLP [8];
- a standard transformer [33]; and
- an E(3)-GATr [106].

They share the generative CFM setup, which is currently the leading technique in precision generation of partonic LHC events [8] and calorimeter showers, for the latter using a vision transformer inside the CFM [141]. Our task does not require translation-equivariant representations. Therefore, we do not insert or extract features related to translational equivariance when using the E(3)-GATr, but internally it can use these representations. Details about the implementation and training can be found in the Appendix A.3.

Strictly speaking, the underlying problem is only symmetric under rotations around the beam axis. To not over-constrain the network we include symmetry breaking multivectors as described in Sec. 6.1.3. For the E(3)-GATr, we include the plane orthogonal to the beam axis as a reference multivector. For L-GATr, we also include a time reference multivector, because the distribution is not invariant under boosts along the beam axis. This step is necessary for any Lorentz-equivariant generative network, as it is not possible to construct a normalized density that is invariant under a non-compact group.

To construct an equivariant generator, we have to choose a base distribution $p_{\text{latent}}(x_1)$ that is invariant under the symmetry group. We use gaussian distributions in the coordinates $(p_x, p_y, p_z, \log m^2)$ with mean and standard deviation fitted to the phase space distribution $p_{\text{data}}(x_0)$. Furthermore, we apply rejection sampling to enforce the phase space constraints $p_T > 22\,\text{GeV}, \Delta R > 0.5$ already at the level of the base distribution.

The phase space parametrization for which we require straight trajectories is crucial for the performance of the generator. The standard MLP and transformer CFMs work directly on x defined as

$$p = \begin{pmatrix} E \\ p_x \\ p_y \\ p_z \end{pmatrix} \to f^{-1}(p) = x = \begin{pmatrix} x_p \\ x_m \\ x_\eta \\ x_\phi \end{pmatrix} \equiv \begin{pmatrix} \log(p_T - p_T^{\min}) \\ \log m^2 \\ \eta \\ \phi \end{pmatrix}, \tag{6.32}$$

to encode $v(x(t), t)$. We standardize all four x-coordinates using their mean and standard deviation over the full dataset. The azimuthal angle ϕ is periodic, and we use this property by adding multiples of 2π to map generated angles into the allowed region $\phi \in [-\pi, \pi]$. We then choose the smallest distance between pairs (x_0, x_1) to construct the target velocity field, allowing paths to cross the boundary at $\phi = \pm\pi$.

In Figure 6.5, we show target probability paths generated in this way. Our approach ensures that none of the trajectories pass through the phase-space region $p_T < p_T^{\min}$, where the target density does not have support; instead, the geodesics lead around this problematic region. The effect of this choice is also displayed in Table 6.6, where the naive choice of straight trajectories in p, corresponding to straight lines in Figure 6.5, gives significantly worse results.

L-GATr starts with p and applies the transformation visualized in Fig. 6.4: first, we use the mapping f to transform x into the corresponding 4-momentum $p = f(x)$. Second, we apply the L-GATr network to obtain the velocity $v_p = \text{L-GATr}(p) = (v_E, v_{p_x}, v_{p_y}, v_{p_z})$ in Minkowski space. Finally, we transform this velocity v_p back into the parametrization x using the jacobian of the backwards transformation, yielding the transformed velocity $v_x = (v_{x_p}, v_{x_m}, v_\eta, v_\phi)$

$$v_x(x(t), t) = \frac{\partial f^{-1}(p)}{\partial p} v_p(p(t), t). \tag{6.33}$$

In practice, we encounter large jacobians from the logarithm transformations in the above relation for v_{x_m}, v_{x_p} due to small values of the jet mass m and the relative transverse momentum $p_T - p_{T,\min}$, leading to unstable training. To avoid this obstacle, we add a fourth step to the procedure, where we overwrite the two problematic velocity components with scalar

Data	Architecture	Base distribution	Periodic	Neg. log-likelihood	AUC
p	L-GATr	rejection sampling	✓	-30.80 ± 0.17	0.945 ± 0.004
x	MLP	rejection sampling	✓	-32.13 ± 0.05	0.780 ± 0.003
x	L-GATr	rejection sampling	✗	-32.57 ± 0.05	0.530 ± 0.017
x	L-GATr	no rejection sampling	✓	-32.58 ± 0.04	0.523 ± 0.014
x	L-GATr	rejection sampling	✓	-32.65 ± 0.04	0.515 ± 0.009

Table 6.6: Impact of the choice of trajectory on different L-GATr networks for the CFM velocity, compared to a MLP velocity network. All networks are trained on the $t\bar{t} + 0j$ dataset.

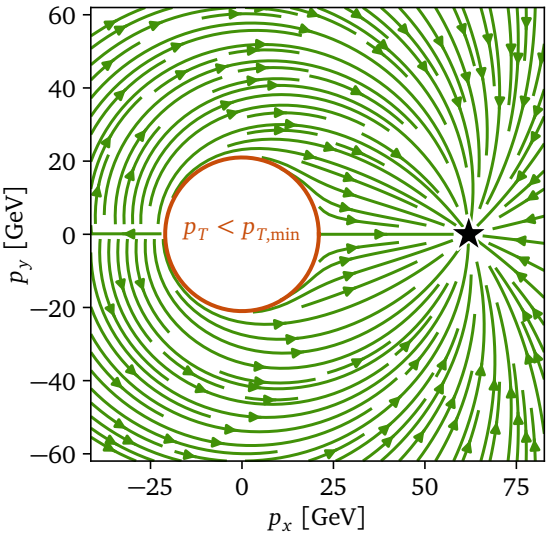

Figure 6.5: Target vector field for Riemannian flow matching. Our choice of metric space guarantees that the generative model respects phase-space boundaries (red circle).

outputs of the L-GATr network. This adds an additional redundant source of symmetry breaking to the reference multivectors discussed above.

For the E(3)-GATr, we encode (p_x, p_y, p_z) as a vector and x_m as a scalar. We then apply a similar transformation as for L-GATr, but without changing the x_m component.

For the full L-GATr architecture we first study the effect of the choice of data representation, of base distribution, and of trajectories in Tab. 6.6.

Performance

We compare a set of 1-dimensional distributions from the different generators in Fig. 6.6. Observables like $p_{T,b}$, that are part of the phase space parametrization Eq. (6.32), are easily learned by all networks because of our choice of target trajectories in Eq. (6.29). All other observables appear as correlations and are harder to learn. L-GATr outperforms the baselines across all distributions. Especially angular correlations benefit from the equivariance encoded in the L-GATr architecture, enabling percent-level precision in these variables for the first time. The main weakness of all architectures are the intermediate top mass poles, requiring the correlation of three external 4-vectors.

To analyze scaling properties, we use the negative log-likelihood evaluated on the events and the AUC of a neural classifier. In Fig. 6.7 we find a clear performance increase with increasing symmetry awareness, from the unstructured MLP over the permutation-equivariant transformer to the rotation-equivariant GATr and the Lorentz-equivariant L-GATr. In particular, the superior L-GATr performance mainly originates from boost-equivariance, as the rotation-equivariant E(3)-GATr performs only

marginally better than the plain transformer. This might come as a surprise, as we allow L-GATr to break this boost equivariance using reference multivectors. This implies that enforcing equivariance in the architecture and then allowing the network to break it with reference multivectors outperforms standard non-equivariant networks. Finally, we also compare to JetGPT [8] approach which constructs the density in a fundamentally different approach. We find that this approach needs more training data than the flow matching models to perform well, but eventually matches the performance of the non-equivariant Transformer CFM and even surpasses it.

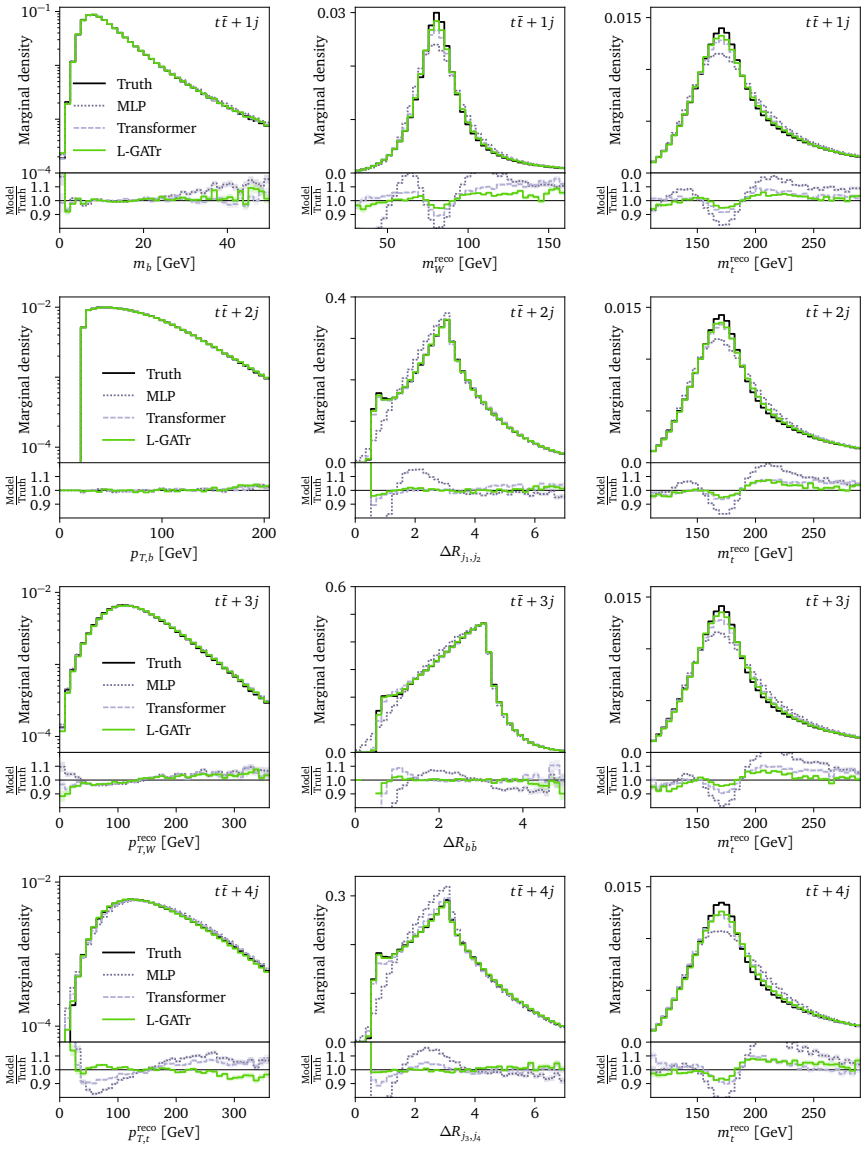

Figure 6.6: Marginal distributions for $t\bar{t}+1,2,3,4$ jets (top to bottom). We do not show E(3)-GATr results, as they are very similar to the standard transformer. The three panels in the bottom row are also included in Ref. [95].

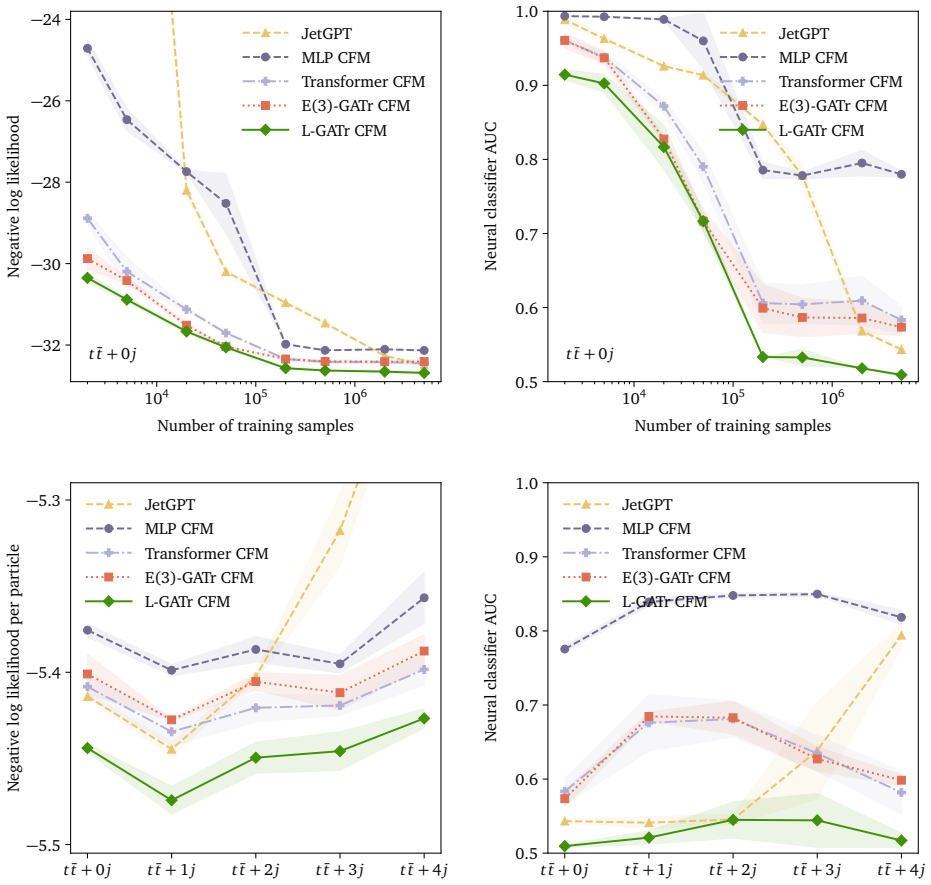

Figure 6.7: Performance of generative networks in terms of a negative log-likelihood over the events (left) and a trained classifier AUC (right). In the top row we show the scaling with the size of the training dataset for the mixed $t\bar{t}+n$ jet dataset, in the bottom row the scaling with the number of particles in the final state. The MLP, Transformer and L-GATr metrics were already discussed in Ref. [95].

6.5 OUTLOOK

Modern ML at the LHC has developed from mostly concept development to the first applications in experiment and theory. For these applications, performance is the main goal, leading us to the question how we can train the most precise neural networks on a large, but nevertheless limited training dataset. In LHC physics, we are in the lucky situation that we can use the known structure of the phase space. It rests on a complex system of symmetries, the leading one being the Lorentz symmetry.

To help our network training, we can encode the Lorentz symmetry or Minkowski metric into the network architecture to avoid learning it. An appropriate internal or latent representation of the Lorentz group then enhances the performance of, essentially, every ML-application working on relativistic phase space objects. Crucially, in cases where symmetries are not

exact, we can allow an equivariant network to break them using symmetry-breaking reference frames, leading to significantly better performance than removing the corresponding equivariance from the network altogether.

L-GATr is a versatile equivariant transformer that constructs such a Lorentz representation for regression, classification, and generation networks. For amplitude regression, a key step in speeding up loop amplitudes in event generators, L-GATr shows the best performance for more than three particles in the final state, thanks to its superior data efficiency, leading to an improved scaling with the phase space dimensionality. For subjet tagging, L-GATr combines the benefit of equivariance with pre-training on large datasets and is at least on par with the best available subjet tagger. Finally, the combination of L-GATr with CFM generator faithfully reproduces the phase space distribution of top pair production with up to four jets better than all other CFM setups. We look forward to further applications of L-GATr at the LHC, as well as generalizations to incorporate additional domain knowledge from collider physics.

7 SUMMARY AND OUTLOOK

Out of all areas of science, high-energy physics is a strong contender for the field in which symmetries play the most central role. Surprisingly, while particle physicists were quick to embrace machine learning, architectures tailored to the symmetries inherent in particle physics problems have received comparably little attention.

Modern architectures that are both scalable and equivariant with respect to the symmetries common in high-energy physics have been explored in this thesis. For permutation equivariance, transformers – already highly optimized for language processing – can be straightforwardly adapted to collider applications. For Lorentz equivariance, a variety of architectures had been proposed, but no transformer offering the favorable scaling properties of that family was available. Consequently, the first Lorentz-equivariant transformer was constructed as a versatile architecture based on geometric-algebra representations.

Transformers are regarded as the leading permutation-equivariant architecture in terms of scaling. Their success in language modelling, notably through autoregressive GPT models, motivated the adaptation of an autoregressive transformer for LHC event generation. In Chapter 4, the adapted model was found to outperform established approaches such as GANs and normalizing flows and to match the performance of denoising diffusion models and conditional flow matching. A Bayesian-neural-network (BNN) variant was developed to permit uncertainty quantification. The method was then extended to generate sequences of variable length in Chapter 5; the resulting autoregressive density was shown to reproduce naturally the hierarchical structure of QCD jet radiation. Techniques allowing extrapolation beyond the training data were investigated, and the BNN uncertainties were observed to capture such out-of-domain regions conservatively.

The Lorentz-Equivariant Geometric Algebra Transformer (L-GATr) architecture was introduced in Chapter 6 as a universal tool for collider physics. Lorentz equivariance is achieved with custom layers built from geometric-algebra representations, combined in a manner analogous to the standard transformer. L-GATr was benchmarked on classification, regression and generative tasks and was found to match or surpass existing architectures in all cases. In particular, the first Lorentz-equivariant generative network was presented.

As the LHC prepares for its high-luminosity programme and legacy measurements, all stages of the analysis pipeline are being optimized. The deployment of performant and data-efficient architectures such as L-GATr could enhance this pipeline at multiple points, with the ultimate

goal of enabling even more precise measurements of nature at its most fundamental level.

A

HYPERPARAMETERS

A.1 LHC EVENT GENERATION WITH JETGPT

hyperparameter	toy models	LHC events
# Gaussians m	21	43
# Bins m	64	-
# TransformerDecoder N	4	4
# Self-attention Heads	4	4
Latent Space Size d	64	128
# Model Parameters	220k	900k
LR Scheduling	one-cycle	one-cycle
Starter LR	3×10^{-4}	10^{-4}
Maximum LR	3×10^{-3}	10^{-3}
Epochs	200	2000
Batch Size	1024	1024
RAdam ϵ	10^{-8}	10^{-4}
# Training Events	600k	2.4M, 670k, 190k
# Generated Events	600k	1M, 1M, 1M

Table A.1: Training setup and hyperparameters for the Bayesian autoregressive transformer.

A.2 EXTRAPOLATING JET RADIATION WITH AUTOREGRESSIVE TRANSFORMERS

Parameter	Value
Optimizer	Adam
Learning rate	$3 \cdot 10^{-4}$
LR schedule	constant
Batch size	512
# Iterations	200k
# Transformer Blocks	3+3
Latent space size d	128
# Attention heads	8
# Mixture model elements	42
# Trainable parameters	1.2M

Table A.2: Architecture and training hyperparameters. We use 3 blocks each for the particle-level transformer and the component-level transformer.

A.3 A LORENTZ-EQUIVARIANT TRANSFORMER FOR ALL OF THE LHC

Amplitude regression

Concerning the DSI baseline, it is an architecture based on the Deep Sets framework [118] that incorporates momentum invariants as part of the input. It works in three stages. First, it applies a different learnable preprocessing block to each particle type in the events, generating a set of latent space representations for each of the particle inputs. Those latent space points are then combined way by summing over all identical particle types, effectively imposing permutation invariance. Finally, the resulting aggregation together with a collection of all momentum invariants of the process is fed to another block that performs the actual regression task. This setup achieves a combination of Lorentz and permutation invariants in an imperfect way.

We list the hyperparameters of all studied baselines in Table A.3. As for the preprocessing, in the case of GAP and L-GATr we standardize the 4-momentum inputs using a common normalization for each component to preserve Lorentz equivariance. For the rest of the baselines we perform ordinary standarization.

Jet Tagging

We provide the L-GATr hyperparamters for top tagging without pre-training in Table A.4. All inputs are preprocessed with a 20 GeV scale factor. L-GATr is trained by minimizing a binary cross entropy loss on the class labels.

Pre-training and full training on JetClass is performed by training L-GATr on the full 100M events over 10^6 iterations. The L-GATr architecture and training hyperparameters is the same that we used for the ordinary top tagging. The only differences are that we now work with 10 output

Hyperparameter	MLP	DSI	Transformer	GAP	CGENN	L-GATr
Architecture	128 channels 5 layers	128 channels 4 layers	128 channels 8 heads 8 blocks	96 scalar ch. 96 multivector ch. 8 blocks	72 scalar ch. 8 multivector ch. 4 blocks	32 scalar ch. 32 multivector ch. 8 heads 8 blocks
Activation	GELU	GELU	GELU	Gated GELU	Gated SiLU [111]	Gated GELU
Parameters	7×10^4	2.6×10^5	1.3×10^6	2.5×10^6	3.2×10^5	1.8×10^6
Optimizer	Adam [142]	Adam [142]	Adam [142]	Adam [142]	Adam [142]	Adam [142]
Learning rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Batch size	256	256	256	256	256	256
Scheduler	-	-	-	-	-	-
Patience	100	100	100	100	100	100
Iterations	2.5×10^6	2.5×10^6	10^6	2.5×10^5	2.5×10^5	2.5×10^5

Table A.3: Hyperparameter summary for all baselines studied for the amplitude task. In the case of DSI, the number of layers and channels refers to both network blocks, and the latent space of each particle has a dimensionality of 64. In the case of CGENN, the hidden node features are identified with the scalar channels, whereas the hidden edge features are identified with the multivector channels.

Hyperparameter	Value
Scalar channels	32
Multivector channels	16
Attention heads	8
Blocks	12
Parameters	1.1×10^6
Optimizer	Lion [52]
Learning rate	3×10^{-4}
Batch size	128
Scheduler	CosineAnnealingLR [143]
Weight decay	0.2
Iterations	2×10^5

Table A.4: Hyperparameter summary for the L-GATr network used for top tagging.

channels and train on a cross entropy loss to accommodate multiclass training, and we use a batch size of 512 to maximize dataset exposure. As for the inputs, the 4-momenta are again scaled by the 20 GeV scale, and the kinematic functions are standardized following the prescription presented in Ref. [29].

Fine-tuning is implemented by resetting the output layer of the pre-trained network and restricting it to one output channel. With this build, the pre-trained weights are trained with a learning rate of 3×10^{-5} and the new weights are trained with a learning rate of 3×10^{-3}. We also apply a weight decay of 0.01 and a batch size of 128. The training is performed across 10^5 iterations.

Event Generation

We summarize the architecture and training hyperparameters of all four generator baselines in Table A.5. We split each dataset into 98% for training and 1% each for validation and testing.

In the classifier test, we train an MLP classifier using binary cross-entropy to distinguish generated events from true events. The classifier inputs include the complete events in the x representation defined in Eq. (6.32), augmented by all pairwise ΔR features, as well as the x representations of the reconstructed particles $t, \bar{t}, W^+, W^-$. The classifier network consists of 3 layers with 256 channels each. Training is conducted over 500 epochs with a batch size of 1024, a dropout rate of 0.1, and the Adam optimizer with default parameters. We start with an initial learning rate of 0.0003, reducing it by a factor of 10 if validation loss shows no improvement for 5 consecutive epochs. The dataset comprises the full truth data and 1M generated events, with an 80%/10%/10% split for training, testing, and validation, respectively.

Hyperparameter	MLP	Transformer	L-GATr and E(3)-GATr
Architecture	336 6 layers	108 channels 6 layers 8 heads	32 scalar ch. 16 multivector ch. 8 heads 6 blocks
Activation	GELU	GELU	Gated GELU
Parameters	5.9×10^5	5.7×10^5	5.4×10^5
Optimizer	Adam [142]	Adam [142]	Adam [142]
Learning rate	10^{-3}	10^{-3}	10^{-3}
Batch size	2048	2048	2048
Iterations	2×10^5	2×10^5	2×10^5

Table A.5: Hyperparameter summary for all baselines studied for the generation task. For all networks, we evaluate the validation loss every 10^3 iterations and decrease the learning rate by a factor of 10 after no improvements for 20 validation steps.

ACKNOWLEDGMENTS

First of all, I'm really grateful to my advisor, Tilman Plehn, for taking me on as a PhD student and for all his guidance over the past three years. I want to thank him for creating such a great research environment in his group, and more broadly, for helping build a strong community for people working at the intersection of physics and machine learning in Heidelberg.

I'd also like to thank Ullrich Köthe for serving as a referee for this thesis, and Stephanie Hansmann-Menzemer and Björn Malte Schäfer for being part of my examination committee.

I am grateful to the graduate school "Particle physics beyond the Standard Model", the CRC TRR 257, and the project "Model-Based AI: Physical Models and Deep Learning for Imaging and Cancer Treatment' of the Carl-Zeiss-Stiftung for funding the first, second and the third year of my PhD, respectively.

I'm grateful to everyone I've had the chance to collaborate with over the past three years. Special thanks go to Johann Brehmer for introducing me to the world of geometric deep learning, and for teaching me how to code and how to present research properly; to Víctor Bresó for the great collaboration that made L-GATr happen; and to Sofia Palacios Schweitzer and Nathan Huetsch for the fun and insightful time exploring the world of HEP-ML together. I'm also thankful to many other collaborators for the successful and enjoyable work we've done together: Pim de Haan, Jesse Thaler, Huilin Qu, Ayodele Ore, Javier Villadamigo, François Charton, Maeve Madigan, Nathanael Ediger, Sebastian Pitz, Luigi Favaro, Peter Lippmann, Gerrit Gerhartz, Fred Hamprecht, Robert Ziegler, Claudio Manzari, Jorge Camalich, Anja Butter, and Peter Sorrenson.

But most importantly, I want to thank all the current and former members of the Heidelberg HEP-ML group for making the past three years such a great experience! In no particular order, I'd like to thank Luigi Favaro, Theo Heimel, Lorenz Vogel, Tanmoy Modak, Ayo Ore, Víctor Bresó, Michel Luchmann, Emma Geoffray, Jona Ackerschott, Claudius Krause, Florian Ernst, Maeve Madigan, Giovanni de Crescenzo, Javier Villadamigo, Nikita Schmal, Antoine Petitjean, Daniel Schiller, Sophia Vent, Sebastian Pitz, Nathanael Ediger, Paula Schuchard, Rebecca Revelli, Henning Bahl, and anyone else I may have unintentionally left out.

Finally, I'm deeply grateful to my friends and family for their constant support over the years. I truly couldn't have done this without you!

[1] C. D. Centre, *Key facts and figures*.

[2] T. Plehn et al., "Modern Machine Learning for LHC Physicists," (2022), arXiv:2211.01421 [hep-ph].

[3] J. Brehmer et al., "Constraining Effective Field Theories with Machine Learning," Phys. Rev. Lett. **121**, 111801 (2018), arXiv:1805.00013 [hep-ph].

[4] G. Kasieczka et al., "The machine learning landscape of top taggers," SciPost Physics **7**, 014 (2019).

[5] ATLAS collaboration, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC," Physics Letters B **716**, 1–29 (2012).

[6] CMS collaboration, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC," Physics Letters B **716**, 30–61 (2012).

[7] L. de Oliveira et al., "Jet-images—deep learning edition," Journal of High Energy Physics **2016**, 1–32 (2016).

[8] A. Butter et al., "Jet diffusion versus JetGPT – Modern networks for the LHC," SciPost Phys. Core **8**, 026 (2025), arXiv:2305.10475 [hep-ph].

[9] A. Einstein, "Zur Elektrodynamik bewegter Körper," Annalen der Physik **4** (1905).

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

[11] E. Bothmann et al., "Event Generation with Sherpa 2.2," SciPost Phys. **7**, 034 (2019), arXiv:1905.09127 [hep-ph].

[12] G. P. Lepage, "Adaptive multidimensional integration: VEGAS enhanced," J. Comput. Phys. **439**, 110386 (2021), arXiv:2009.05112 [physics.comp-ph].

[13] G. Altarelli and G. Parisi, "Asymptotic Freedom in Parton Language," Nucl. Phys. B **126**, 298–318 (1977).

[14] B. Andersson, G. Gustafson, and B. Soderberg, "A General Model for Jet Fragmentation," Z. Phys. C **20**, 317 (1983).

[15] T. Sjostrand, "Jet Fragmentation of Nearby Partons," Nucl. Phys. B **248**, 469–502 (1984).

[16] J.-C. Winter, F. Krauss, and G. Soff, "A Modified cluster hadronization model," Eur. Phys. J. C **36**, 381–395 (2004), arXiv:hep-ph/0311085.

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

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

[19] A. M. Sirunyan et al., "Particle-flow reconstruction and global event description with the CMS detector," JINST **12**, P10003 (2017), arXiv:1706.04965 [physics.ins-det].

[20] M. Aaboud et al., "Jet reconstruction and performance using particle flow with the ATLAS Detector," Eur. Phys. J. C **77**, 466 (2017), arXiv:1703.10485 [hep-ex].

[21] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-k_t jet clustering algorithm," JHEP **04**, 063 (2008), arXiv:0802.1189 [hep-ph].

[22] E. Bothmann et al., "Exploring phase space with Neural Importance Sampling," SciPost Phys. **8**, 069 (2020), arXiv:2001.05478 [hep-ph].

[23] T. Heimel et al., "MadNIS - Neural multi-channel importance sampling," SciPost Phys. **15**, 141 (2023), arXiv:2212.06172 [hep-ph].

[24] T. Heimel et al., "The MadNIS reloaded," SciPost Phys. **17**, 023 (2024), arXiv:2311.01548 [hep-ph].

[25] C. Bierlich et al., "Towards a data-driven model of hadronization using normalizing flows," SciPost Phys. **17**, 045 (2024), arXiv:2311.09296 [hep-ph].

[26] J. Chan et al., "Fitting a deep generative hadronization model," JHEP **09**, 084 (2023), arXiv:2305.17169 [hep-ph].

[27] L. Ehrke et al., "Topological reconstruction of particle physics processes using graph neural networks," Phys. Rev. D **107**, 116019 (2023), arXiv:2303.13937 [hep-ph].

[28] A. Butter et al., "The Machine Learning landscape of top taggers," SciPost Phys. **7**, edited by G. Kasieczka and T. Plehn, 014 (2019), arXiv:1902.09914 [hep-ph].

[29] H. Qu, C. Li, and S. Qian, "Particle Transformer for Jet Tagging," (2022), arXiv:2202.03772 [hep-ph].

[30] J. Brehmer et al., "A Guide to Constraining Effective Field Theories with Machine Learning," Phys. Rev. **D98**, 052004 (2018), arXiv:1805.00020 [hep-ph].

[31] A. Andreassen et al., "OmniFold: A Method to Simultaneously Unfold All Observables," Phys. Rev. Lett. **124**, 182001 (2020), arXiv:1911.09107 [hep-ph].

[32] M. Bellagente et al., "Invertible Networks or Partons to Detector and Back Again," SciPost Phys. **9**, 074 (2020), arXiv:2006.06685 [hep-ph].

[33] N. Huetsch, J. Mariño, et al., "The Landscape of Unfolding with Machine Learning," (2024), arXiv:2404.18807 [hep-ph].

[34] A. Butter et al., "Two invertible networks for the matrix element method," SciPost Phys. **15**, 094 (2023), arXiv:2210.00019 [hep-ph].

[35]T. Heimel et al., "Precision-Machine Learning for the Matrix Element Method," (2023), arXiv:2310.07752 [hep-ph].

[36]G. Kasieczka et al., "The LHC Olympics 2020 a community challenge for anomaly detection in high energy physics," Rept. Prog. Phys. **84**, 124201 (2021), arXiv:2101.08320 [hep-ph].

[37]A. Butter et al., "GANplifying event samples," SciPost Phys. **10**, 139 (2021), arXiv:2008.06545 [hep-ph].

[38]S. Bieringer et al., "Calomplification — the power of generative calorimeter models," JINST **17**, P09028 (2022), arXiv:2202.07352 [hep-ph].

[39]S. Badger et al., "Loop amplitudes from precision networks," SciPost Phys. Core **6**, 034 (2023), arXiv:2206.14831 [hep-ph].

[40]D. Maître and H. Truong, "A factorisation-aware Matrix element emulator," JHEP **11**, 066 (2021), arXiv:2107.06625 [hep-ph].

[41]C. Krause et al., "CaloChallenge 2022: A Community Challenge for Fast Calorimeter Simulation," (2024), arXiv:2410.21611 [cs.LG].

[42]A. Butter, T. Plehn, and R. Winterhalder, "How to GAN LHC Events," SciPost Phys. **7**, 075 (2019), arXiv:1907.03764 [hep-ph].

[43]A. Butter et al., "Generative networks for precision enthusiasts," SciPost Phys. **14**, 078 (2023), arXiv:2110.13632 [hep-ph].

[44]A. Einstein, "Zur Elektrodynamik bewegter Körper," Annalen Phys. **322**, 891–921 (1905).

[45]C. S. Wu et al., "Experimental Test of Parity Conservation in β Decay," Phys. Rev. **105**, 1413–1414 (1957).

[46]J. H. Christenson et al., "Evidence for the 2π Decay of the K_2^0 Meson," Phys. Rev. Lett. **13**, 138–140 (1964).

[47]K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural networks **2**, 359–366 (1989).

[48]A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," (2019), arXiv:1912.01703 [cs.LG].

[49]A. Karpathy, *3e-4 is the best learning rate for Adam, hands down.* Tweet, Twitter, 24 November 2016.

[50]D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," (2014), arXiv:1412.6980 [cs.LG].

[51]L. Liu et al., "On the variance of the adaptive learning rate and beyond," (2019), arXiv:1908.03265 [cs.LG].

[52]X. Chen et al., "Symbolic discovery of optimization algorithms," (2023), arXiv:2302.06675 [cs.LG].

[53]D. MacKay, "Probable Networks and Plausible Predictions – A Review of Practical Bayesian Methods for Supervised Neural Networks," Comp. in Neural Systems **6**, 4679 (1995).

[54] R. M. Neal, "Bayesian learning for neural networks," PhD thesis (Toronto, 1995).

[55] Y. Gal, "Uncertainty in Deep Learning," PhD thesis (Cambridge, 2016).

[56] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" Proc. NIPS (2017), arXiv:1703.04977 [cs.CV].

[57] M. M. Bronstein et al., "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges," (2021), arXiv:2104.13478 [cs.LG].

[58] A. Vaswani et al., "Attention is all you need," Advances in neural information processing systems **30** (2017), arXiv:1706.03762 [cs.CL].

[59] T. Dao et al., "FlashAttention: fast and memory-efficient exact attention with IO-awareness," in Advances in neural information processing systems (neurips) (2022).

[60] T. Dao, "FlashAttention-2: faster attention with better parallelism and work partitioning," in International conference on learning representations (iclr) (2024).

[61] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," (2016), arXiv:1607.06450 [stat.ML].

[62] K. He et al., "Deep Residual Learning for Image Recognition," 10.1109/CVPR.2016.90 (2015), arXiv:1512.03385 [cs.CV].

[63] A. Dosovitskiy et al., "An image is worth 16x16 words: transformers for image recognition at scale," ICLR (2021), eprint: 2010.11929 (cs.CV).

[64] A. Butter et al., "Jet diffusion versus jetgpt–modern networks for the lhc," arXiv preprint arXiv:2305.10475 (2023).

[65] A. Radford et al., "Language models are unsupervised multitask learners," OpenAI blog **1**, 9 (2019).

[66] M. Bellagente et al., "Understanding Event-Generation Networks via Uncertainties," SciPost Phys. **13**, 003 (2022), arXiv:2104.04543 [hep-ph].

[67] S. Catani et al., "QCD matrix elements + parton showers," JHEP **11**, 063 (2001), arXiv:hep-ph/0109231.

[68] M. Cacciari, G. P. Salam, and G. Soyez, "FastJet User Manual," Eur. Phys. J. C **72**, 1896 (2012), arXiv:1111.6097 [hep-ph].

[69] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-k_t jet clustering algorithm," JHEP **04**, 063 (2008), arXiv:0802.1189 [hep-ph].

[70] S. Badger et al., "Machine learning and LHC event generation," SciPost Phys. **14**, edited by A. Butter, T. Plehn, and S. Schumann, 079 (2023), arXiv:2203.07460 [hep-ph].

[71] A. Butter et al., "Extrapolating Jet Radiation with Autoregressive Transformers," Submitted to SciPost Phys. (2024), arXiv:2412.12074 [hep-ph].

[72] R. K. Ellis, W. J. Stirling, and B. R. Webber, *QCD and collider physics*, Vol. 8 (Cambridge University Press, Feb. 2011).

[73]T. Plehn, "Lectures on LHC Physics," Lect. Notes Phys. **844**, 1–193 (2012), arXiv:0910.4182 [hep-ph].

[74]J. Campbell, J. Huston, and F. Krauss, *The Black Book of Quantum Chromodynamics : a Primer for the LHC Era* (Oxford University Press, 2018).

[75]M. van Beekveld et al., "PanScales showers for hadron collisions: all-order validation," JHEP **11**, 020 (2022), arXiv:2207.09467 [hep-ph].

[76]A. Andreassen et al., "JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics," Eur. Phys. J. **C79**, 102 (2019), arXiv:1804.09720 [hep-ph].

[77]A. Andreassen et al., "Binary JUNIPR: an interpretable probabilistic model for discrimination," Phys. Rev. Lett. **123**, 182001 (2019), arXiv:1906.10137 [hep-ph].

[78]T. Finke et al., "Learning the language of QCD jets with transformers," (2023), arXiv:2303.07364 [hep-ph].

[79]J. Bellm et al., "Herwig 7.0/Herwig++ 3.0 release note," Eur. Phys. J. **C76**, 196 (2016), arXiv:1512.01178 [hep-ph].

[80]T. Sjöstrand et al., "An Introduction to PYTHIA 8.2," Comput. Phys. Commun. **191**, 159–177 (2015), arXiv:1410.3012 [hep-ph].

[81]E. Bothmann et al., "Event generation with Sherpa 3," (2024), arXiv:2410.22148 [hep-ph].

[82]M. L. Mangano et al., "ALPGEN, a generator for hard multiparton processes in hadronic collisions," JHEP **07**, 001 (2003), arXiv:hep-ph/0206293.

[83]S. Frixione, P. Nason, and C. Oleari, "Matching NLO QCD computations with Parton Shower simulations: the POWHEG method," JHEP **11**, 070 (2007), arXiv:0709.2092 [hep-ph].

[84]E. Bothmann et al., "Efficient precision simulation of processes with many-jet final states at the LHC," Phys. Rev. D **109**, 014013 (2024), arXiv:2309.13154 [hep-ph].

[85]S. Höche, F. Krauss, and D. Reichelt, "The Alaric parton shower for hadron colliders," (2024), arXiv:2404.14360 [hep-ph].

[86]M. R. Buckley, T. Plehn, and M. J. Ramsey-Musolf, "Top squark with mass close to the top quark," Phys. Rev. D **90**, 014046 (2014), arXiv:1403.2726 [hep-ph].

[87]M. van Beekveld et al., "Introduction to the PanScales framework, version 0.1," SciPost Phys. Codeb. **2024**, 31 (2024), arXiv:2312.13275 [hep-ph].

[88]M. van Beekveld et al., "A new standard for the logarithmic accuracy of parton showers," (2024), arXiv:2406.02661 [hep-ph].

[89]S. D. Ellis, R. Kleiss, and W. J. Stirling, "W's, Z's and Jets," Phys. Lett. B **154**, 435–440 (1985).

[90]F. A. Berends et al., "Multi - Jet Production in W, Z Events at $p\bar{p}$ Colliders," Phys. Lett. B **224**, 237–242 (1989).

[91]F. A. Berends et al., "On the production of a W and jets at hadron colliders," Nucl. Phys. B **357**, 32–64 (1991).

[92]E. Gerwick, T. Plehn, and S. Schumann, "Understanding Jet Scaling and Jet Vetos in Higgs Searches," Phys. Rev. Lett. **108**, 032003 (2012), arXiv:1108.3335 [hep-ph].

[93]E. Gerwick et al., "Scaling Patterns for QCD Jets," JHEP **10**, 162 (2012), arXiv:1208.3676 [hep-ph].

[94]J. Ackerschott et al., "Returning CP-observables to the frames they belong," SciPost Phys. **17**, 001 (2024), arXiv:2308.00027 [hep-ph].

[95]J. Spinner et al., "Lorentz-equivariant geometric algebra transformers for high-energy physics," Advances in Neural Information Processing Systems (2024), arXiv:2405.14806 [physics.data-an].

[96]T. Golling et al., "Masked particle modeling on sets: towards self-supervised high energy physics foundation models," Mach. Learn. Sci. Tech. **5**, 035074 (2024), arXiv:2401.13537 [hep-ph].

[97]M. Leigh et al., "Is Tokenization Needed for Masked Particle Modelling?" (2024), arXiv:2409.12589 [hep-ph].

[98]J. Birk, A. Hallin, and G. Kasieczka, "OmniJet-α: the first cross-task foundation model for particle physics," Mach. Learn. Sci. Tech. **5**, 035031 (2024), arXiv:2403.05618 [hep-ph].

[99]G. Kasieczka et al., "Per-Object Systematics using Deep-Learned Calibration," SciPost Phys. **9**, 089 (2020), arXiv:2003.11099 [hep-ph].

[100]G. Aad et al., "Precision calibration of calorimeter signals in the ATLAS experiment using an uncertainty-aware neural network," (2024), arXiv:2412.04370 [hep-ex].

[101]S. Bollweg et al., "Deep-Learning Jets with Uncertainties and More," SciPost Phys. **8**, 006 (2020), arXiv:1904.10004 [hep-ph].

[102]S. Bieringer et al., "Calibrating Bayesian generative machine learning for Bayesiamplification," Mach. Learn. Sci. Tech. **5**, 045044 (2024), arXiv:2408.00838 [cs.LG].

[103]J. Brehmer et al., "A Lorentz-Equivariant Transformer for All of the LHC," Submitted to SciPost Phys. (2024), arXiv:2411.00446 [hep-ph].

[104]D. Hestenes, *Space-time algebra*, Documents on modern physics (Gordon and Breach, 1966).

[105]S. Villar et al., "Scalars are universal: gauge-equivariant machine learning, structured like classical physics," CoRR (2021), arXiv:2106.06610 [cs.LG].

[106]J. Brehmer et al., "Geometric Algebra Transformer," in Advances in Neural Information Processing Systems, Vol. 37, edited by H. Larochelle et al. (May 2023), arXiv:2305.18415 [cs.LG].

[107] R. Xiong et al., "On layer normalization in the transformer architecture," in International Conference on Machine Learning (PMLR, 2020), pp. 10524–10533, arXiv:2002.04745 [cs.LG].

[108] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," (2016), arXiv:1606.08415 [cs.LG].

[109] D. Maître, V. S. Ngairangbam, and M. Spannowsky, "Optimal Equivariant Architectures from the Symmetries of Matrix-Element Likelihoods," (2024), arXiv:2410.18553 [hep-ph].

[110] A. Bogatskiy et al., "Explainable equivariant neural networks for particle physics: PELICAN," JHEP **03**, 113 (2024), arXiv:2307.16506 [hep-ph].

[111] D. Ruhe, J. Brandstetter, and P. Forré, "Clifford group equivariant neural networks," in Advances in Neural Information Processing Systems, Vol. 37 (May 2023), arXiv:2305.11141 [cs.LG].

[112] S. Gong et al., "An efficient Lorentz equivariant graph neural network for jet tagging," JHEP **07**, 030 (2022), arXiv:2201.08187 [hep-ph].

[113] M. M. Bronstein et al., "Geometric deep learning: grids, groups, graphs, geodesics, and gauges," (2021), arXiv:2104.13478 [cs.LG].

[114] A. Bogatskiy et al., "PELICAN: Permutation Equivariant and Lorentz Invariant or Covariant Aggregator Network for Particle Physics," (2022), arXiv:2211.00454 [hep-ph].

[115] J. Aylett-Bullock, S. Badger, and R. Moodie, "Optimising simulations for diphoton production at hadron colliders using amplitude neural networks," JHEP **08**, 066 (2021), arXiv:2106.09474 [hep-ph].

[116] D. Maître and H. Truong, "One-loop matrix element emulation with factorisation awareness," 10.1007/JHEP05(2023)159 (2023), arXiv:2302.04005 [hep-ph].

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

[118] M. Zaheer et al., "Deep sets," (2017), arXiv:1703.06114 [cs.LG].

[119] T. Heimel et al., "QCD or What?" SciPost Phys. **6**, 030 (2019), arXiv:1808.08979 [hep-ph].

[120] J. de Favereau et al., "DELPHES 3, A modular framework for fast simulation of a generic collider experiment," JHEP **02**, 057 (2014), arXiv:1307.6346 [hep-ex].

[121] M. Aaboud et al., "Performance of top-quark and W-boson tagging with ATLAS in Run 2 of the LHC," Eur. Phys. J. C **79**, 375 (2019), arXiv:1808.07858 [hep-ex].

[122] A. Butter et al., "Deep-learned Top Tagging with a Lorentz Layer," SciPost Phys. **5**, 028 (2018), arXiv:1707.08966 [hep-ph].

[123]L. Moore et al., "Reports of My Demise Are Greatly Exaggerated: N-subjettiness Taggers Take On Jet Images," SciPost Phys. **7**, 036 (2019), arXiv:1807.04769 [hep-ph].

[124]P. T. Komiske, E. M. Metodiev, and J. Thaler, "Energy Flow Networks: Deep Sets for Particle Jets," JHEP **01**, 121 (2019), arXiv:1810.05165 [hep-ph].

[125]G. Louppe et al., "QCD-Aware Recursive Neural Networks for Jet Physics," JHEP **01**, 057 (2019), arXiv:1702.00748 [hep-ph].

[126]H. Qu and L. Gouskos, "ParticleNet: Jet Tagging via Particle Clouds," Phys. Rev. D **101**, 056019 (2020), arXiv:1902.08570 [hep-ph].

[127]Y. Wu et al., "Jet Tagging with More-Interaction Particle Transformer," 10.1088/1674-1137/ad7f3d (2024), arXiv:2407.08682 [hep-ph].

[128]G. Kasieczka et al., *Top quark tagging reference dataset*, https://doi.org/10.5281/zenodo.2603256, Mar. 2019.

[129]J. de Favereau et al., "DELPHES 3, A modular framework for fast simulation of a generic collider experiment," JHEP **02**, 057 (2014), arXiv:1307.6346 [hep-ex].

[130]G. Kasieczka et al., "Deep-learning Top Taggers or The End of QCD?" JHEP **05**, 006 (2017), arXiv:1701.08784 [hep-ph].

[131]M. Bellagente et al., "How to GAN away Detector Effects," SciPost Phys. **8**, 070 (2020), arXiv:1912.00477 [hep-ph].

[132]M. Backes et al., "An unfolding method based on conditional invertible neural networks (cINN) using iterative training," SciPost Phys. Core **7**, 007 (2024), arXiv:2212.08674 [hep-ph].

[133]A. Shmakov et al., "End-To-End Latent Variational Diffusion Models for Inverse Problems in High Energy Physics," (2023), arXiv:2305.10399 [hep-ex].

[134]S. Diefenbacher et al., "Improving generative model-based unfolding with Schrödinger bridges," Phys. Rev. D **109**, 076011 (2024), arXiv:2308.12351 [hep-ph].

[135]J. Alwall et al., "MadGraph 5 : Going Beyond," JHEP **06**, 128 (2011), arXiv:1106.0522 [hep-ph].

[136]M. Cacciari, G. P. Salam, and G. Soyez, "FastJet User Manual," Eur. Phys. J. C **72**, 1896 (2012), arXiv:1111.6097 [hep-ph].

[137]G. Aad et al., "Measurements of top-quark pair single- and double-differential cross-sections in the all-hadronic channel in pp collisions at $\sqrt{s} = 13$ TeV using the ATLAS detector," JHEP **01**, 033 (2021), arXiv:2006.09274 [hep-ex].

[138]R. T. Chen et al., "Neural ordinary differential equations," Advances in Neural Information Processing Systems **31** (2018).

[139]Y. Lipman et al., "Flow matching for generative modeling," (2023), arXiv:2210.02747 [cs.LG].

[140]M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden, "Stochastic interpolants: a unifying framework for flows and diffusions," (2023), arXiv:2303.08797 [cs.LG].

[141]L. Favaro et al., "CaloDREAM – Detector Response Emulation via Attentive flow Matching," (2024), arXiv:2405.09629 [hep-ph].

[142]D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," (2014), arXiv:1412.6980 [cs.LG].

[143]I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with restarts," CoRR (2016), arXiv:1608.03983 [cs.LG].