

Developing Foundation Models for Particle Physics

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

vorgelegt von
Joschka Valentin Maria Birk

Hamburg
2026

Gutachter/innen der Dissertation:

Prof. Dr. Gregor Kasieczka
Prof. Dr. Marcus Brüggem

Zusammensetzung der Prüfungskommission:

Prof. Dr. Gregor Kasieczka
Prof. Dr. Marcus Brüggem
Prof. Dr. Arwen Pearson
Prof. Dr. Sarah Heim
Prof. Dr. Timo Weigand

Vorsitzende der Prüfungskommission:

Prof. Dr. Arwen Pearson

Datum der Disputation:

22. September 2026

Vorsitzender des Fach-Promotionsausschusses PHYSIK:

Prof. Dr. Johannes Haller

Leiter des Fachbereichs PHYSIK:

Prof. Dr. Markus Drescher

Dekan der Fakultät MIN:

Prof. Dr.-Ing. Norbert Ritter

Eidesstattliche Versicherung / Declaration on oath

Hiermit versichere ich an Eides statt, die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt zu haben.

Sofern im Zuge der Erstellung der vorliegenden Dissertationsschrift generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

I hereby declare and affirm that this doctoral dissertation is my own work and that I have not used any aids and sources other than those indicated.

If electronic resources based on generative artificial intelligence (gAI) were used in the course of writing this dissertation, I confirm that my own work was the main and value-adding contribution and that complete documentation of all resources used is available in accordance with good scientific practice. I am responsible for any erroneous or distorted content, incorrect references, violations of data protection and copyright law or plagiarism that may have been generated by the gAI.

Hamburg, den 21.08.2026



Unterschrift der Doktorandin/des Doktoranden

Zusammenfassung

Foundation Models haben das Feld der künstlichen Intelligenz revolutioniert. Über viele Jahre hinweg wurden Machine Learning Modelle jeweils individuell für eine Vielzahl an Anwendungen entwickelt. Foundation Models hingegen basieren auf umfassendem sogenanntem Pre-Training und anschließendem Fine-Tuning, was zu einer erheblichen Verbesserung bei der finalen Anwendung führt. Diese Dissertation stellt die Entwicklung solcher Pre-Training-Methoden zugeschnitten auf die Teilchenphysik dar, wobei der Fokus auf sogenannten Teilchenjets – Bündel kollimierter Teilchen, welche bei hochenergetischen Kollisionsprozessen in Teilchenbeschleunigern wie dem LHC entstehen – gesetzt wird. Die hier vorgelegte Arbeit folgt dem folgenden Aufbau: zunächst wird ein auf dem sogenannte Flow-Matching-Prinzip basierendes generatives Modell für Teilchenjet-Bestandteile vorgestellt. Dieses Modell ist das erste generative Modell für Teilchenjets, welches nicht nur die kinematischen Informationen der Jetbestandteile erzeugt, sondern auch deren Teilchen-ID sowie die Verschiebung ihrer Trajektorie bezüglich des Kollisionspunktes. Die zweite Studie adaptiert die sogenannte Next Token Prediction (NTP), welche das grundlegende Konzept für große Sprachmodelle ist, für Teilchenjets. Daraus geht das erste Cross-Task Foundation Model der Teilchenphysik hervor. Basierend auf dieser initialen Arbeit zu generativem Pre-Training für Teilchenjets mit NTP wird der Vorteil, dass NTP keine externen Label benötigt, voll nutzbar gemacht, indem ein Pre-Training auf gemessenen Daten eines Beschleunigerexperiments durchgeführt wird. Darauf aufbauend werden mehrere Verbesserungen der NTP-basierten Pre-Training-Methode vorgestellt, und die NTP Methode wird mit der sehr ähnlichen Methode des Masked Particle Modeling verglichen. Insbesondere wird hierbei eine Pre-Training-Methode eingeführt, welche diese beiden Ansätze kombiniert, was zu einer deutlichen Verbesserung gegenüber des nur auf NTP basierten Pre-Trainings führt. Schließlich wird eine umfassende Vergleichsstudie gezeigt, welche unterschiedliche Pre-Training-Methoden mit und ohne der Nutzung von externen Labels analysiert. Gemeinsam repräsentieren diese Beiträge mehrere Meilensteine hin zu der Entwicklung leistungsstarker Foundation Models in der Teilchenphysik.

Abstract

Foundation models have revolutionized the field of artificial intelligence. For many years, machine learning models were designed and trained independently for a variety of tasks. Foundation models, on the other hand, rely on powerful pre-training methods that enhance model performance on downstream tasks. This thesis presents the development of such pre-training methods tailored for particle physics, with a particular focus on particle jets – collimated sprays of particles measured in high-energy collision experiments, such as at the LHC. The research follows a progressive trajectory: first, a generative model for particle jet constituents, based on the flow matching paradigm, is introduced. It represents the first model for particle jets capable of generating not only the kinematic information of the jet constituents, but also their particle-ID and trajectory displacement features. The second study adapts next token prediction (NTP) – the cornerstone of large language model pre-training – to particle jets, resulting in the field’s first cross-task foundation model. Following the initial work on generative pre-training for particle jets with NTP, the unsupervised nature of this approach is fully exploited by applying it to measured, unlabeled collider experiment data. Next, several improvements to the NTP-based pre-training strategy are introduced, and the NTP approach is compared to the closely related masked particle modeling pre-training paradigm. Most notably, a novel joint pre-training is introduced that combines both methods, yielding a substantial performance increase compared to pre-training on NTP alone. Finally, a comprehensive comparison of unsupervised and supervised pre-training approaches is presented. Collectively, these contributions represent multiple defining milestones towards the development of powerful foundation models in the field of particle physics.

Contents

Preface	1
1 Introduction	3
I Theory and Methodology	5
2 Particle Physics	7
2.1 The Standard Model of Particle Physics	7
2.1.1 Elementary Particles and Fundamental Interactions	7
2.1.2 The Standard Model as a Quantum Field Theory	8
2.2 Beyond the Standard Model	12
2.3 Collider Physics	14
2.3.1 The Large Hadron Collider	15
2.3.2 Particle Detectors	15
2.3.3 Hadronic Jets	21
3 Machine Learning	27
3.1 Overview	27
3.1.1 Loss Functions	27
3.1.2 Supervised Learning	28
3.1.3 Unsupervised Learning	28
3.2 Model Optimization	29
3.3 Neural Networks	32
3.3.1 Multi-Layer Perceptron	32
3.3.2 Network Architectures for Point Clouds	34
3.4 Generative Modeling	37
3.4.1 Flow Matching	37
3.4.2 Next Token Prediction	39
3.5 Masked Modeling	42
3.6 Tokenization with Vector-Quantized Variational Autoencoders	43
4 Foundation Models for Particle Physics	46
4.1 Foundation Models	46
4.2 Jet Physics as a Testbed for Foundation Models in HEP	47
4.3 R&D Benchmark Datasets	49
4.3.1 Top Tagging Reference Dataset	49
4.3.2 The JetNet Dataset	49
4.3.3 The JetClass Dataset	50
4.4 Pre-Training Methods for Jet Foundation Models	51
II Research Contributions	53
5 Flow Matching Beyond Kinematics [1]	56

CONTENTS

6	OmniJet-α [2]	61
7	Aspen Open Jets [3]	65
8	Enhancing Generative Pre-Training for Jet Foundation Models [4]	69
9	A Comparison of Jet Foundation Model Pre-Training Objectives [5]	74
10	Conclusion	79
	Acknowledgements	81
	References	82
	List of Acronyms	101
	Publications	103
	Flow Matching Beyond Kinematics: Generating Jets with Particle Identification and Trajectory Displacement Information	104
	OmniJet- α : The First Cross-Task Foundation Model for Particle Physics	120
	Aspen Open Jets: Unlocking LHC Data for Foundation Models in Particle Physics	143
	Enhancing Next Token Prediction Based Pre-Training for Jet Foundation Models .	156
	Pre-Training for Simulation-Based Science: A Study on Jet Foundation Model Training Objectives	172

Preface

This cumulative thesis presents the results of my research conducted at the Institute of Experimental Physics at the University of Hamburg from 2023 to 2026 and at Stanford University and SLAC in 2025. The corresponding (pre-)publications are listed in the following. These papers have either been published in peer-reviewed journals, or are currently under review at such journals. The research was conducted in collaboration with other researchers. The results of those (pre-)publications are summarized in chapters 5 to 9. My individual involvement in each work is disclosed at the beginning of those chapters.

- [1] Joschka Birk, Erik Buhmann, Cedric Ewen, Gregor Kasieczka, and David Shih. “Flow matching beyond kinematics: Generating jets with particle identification and trajectory displacement information”. In: *Phys. Rev. D* 111.5 (2025), p. 052008. DOI: 10.1103/PhysRevD.111.052008. arXiv: 2312.00123 [hep-ph].
- [2] Joschka Birk, Anna Hallin, and Gregor Kasieczka. “OmniJet- α : the first cross-task foundation model for particle physics”. In: *Mach. Learn. Sci. Tech.* 5.3 (2024), p. 035031. DOI: 10.1088/2632-2153/ad66ad. arXiv: 2403.05618 [hep-ph].
- [3] Oz Amram, Luca Anzalone, Joschka Birk, Darius A. Faroughy, Anna Hallin, Gregor Kasieczka, Michael Krämer, Ian Pang, Humberto Reyes-Gonzalez, and David Shih. “Aspen Open Jets: unlocking LHC data for foundation models in particle physics”. In: *Mach. Learn. Sci. Tech.* 6.3 (2025), p. 030601. DOI: 10.1088/2632-2153/ade58f. arXiv: 2412.10504 [hep-ph].
- [4] Joschka Birk, Anna Hallin, Gregor Kasieczka, Nikol Madzharova, Ian Pang, and David Shih. “Enhancing next token prediction based pre-training for jet foundation models”. In: *Mach. Learn. Sci. Tech.* 7.3 (2026), p. 035042. DOI: 10.1088/2632-2153/ae70bc. arXiv: 2512.04149 [hep-ph].
- [5] Ibrahim Elsharkawy, Joschka Birk, Vinicius Mikuni, Wahid Bhimji, Gregor Kasieczka, and Benjamin Nachman. “Pre-Training for Simulation-Based Science: A Study on Jet Foundation Model Training Objectives” (June 2026). Submitted to *PRX Intelligence*, in review. arXiv: 2606.14870 [hep-ph].

Additionally, while not discussed in this thesis, I contributed to the following publications:

- [6] Joschka Birk, Frank Gaede, Anna Hallin, Gregor Kasieczka, Martina Mozzanica, and Henning Rose. “OmniJet- α_C : learning point cloud calorimeter simulations using generative transformers”. In: *JINST* 20.07 (2025), P07007. DOI: 10.1088/1748-0221/20/07/P07007. arXiv: 2501.05534 [hep-ph].
- [7] Laurits Tani, Joosep Pata, and Joschka Birk. “Reconstructing hadronically decaying tau leptons with a jet foundation model”. In: *SciPost Phys. Core* 8 (2025), p. 046. DOI: 10.21468/SciPostPhysCore.8.3.046. arXiv: 2503.19165 [hep-ex].
- [8] Joschka Birk, Gregor Kasieczka, Siddharth Mishra-Sharma, Benjamin Nachman, Dennis Noll, and Tanvi Wamorkar. “A Scientific Human-Agent Reproduction Pipeline”. In: *Physics and AI Conference 2026 (PAI2026)*. June 2026. DOI: 10.5281/zenodo.21078068. arXiv: 2604.18752 [hep-ph].

- [9] Joschka Birk, Frank Gaede, Anna Hallin, Gregor Kasieczka, Martina Mozzanica, and Henning Rose. “SPADE: Split-and-Delay Embeddings for Autoregressive High-Granularity Calorimeter Simulation” (June 2026). Submitted to *Mach. Learn. Sci. Tech.* arXiv: 2606.11304 [physics.ins-det].

During the course of my doctoral studies, I co-supervised the following Bachelor’s and Master’s theses:

- [10] Dana Horwitz. “Integrating jet clustering history information in particle tagging”. Bachelor’s thesis. University of Hamburg, 2024.
- [11] Finn Johannsen. “Pre-training strategies for a foundation model in jet physics”. Master’s thesis. University of Hamburg, 2024.
- [12] Henning Rose. “OmniJet- α_C : Learning point cloud calorimeter simulations using generative transformers”. Master’s thesis. University of Hamburg, 2025.
- [13] Nikol Madzharova. “Improving Tokenisation for Foundation Models in Particle Physics”. Master’s thesis. University of Hamburg, 2026.

I used generative AI tools during the research process and while writing this thesis, in accordance with the “declaration on oath”. In particular, I used large language model (LLM) based tools as coding and writing assistants. All AI-generated suggestions were thoroughly reviewed, verified, and, if necessary, modified before being incorporated into the thesis text. The AI tools used are listed below.

- Claude Code:
 - Used to assist with literature research.
 - Used as a coding assistant.
 - Used to suggest language refinements in the thesis text: spell checking; grammar corrections; wording improvements and rephrasing of individual sentences to improve the readability and clarity of the text.
 - Used to assist with the $\text{\LaTeX}$ configuration of this document.
- GitHub Copilot:
 - Used as a coding assistant.
- you.com:
 - Used to assist with literature research.
- Grammarly:
 - Used to suggest language refinements in the thesis text: spell checking; grammar corrections; wording improvements and rephrasing of individual sentences to improve the readability and clarity of the text.

1 Introduction

The field of particle physics, also referred to as high energy physics (HEP), studies the elementary particles of our universe and how those particles interact with each other. Collider physics, which is a subfield of HEP, focuses on investigating the properties of elementary particles by first accelerating particles to extraordinary velocities close to the speed of light and then bringing them to collision. By analyzing those collision events, researchers can study the properties of already known particles, and also search for new, so far unknown particles. The Standard Model (SM) is an extraordinarily successful theory that describes the behavior of the known elementary particles. In 2012, the ATLAS [14] and CMS [15] experiments at the Large Hadron Collider (LHC) [16, 17] at CERN discovered the Higgs boson [18, 19]. The discovery of the Higgs boson marked an important milestone in experimental particle physics, since it provided experimental evidence that the so-called Higgs mechanism [20–25] provides an accurate description of how elementary particles acquire their mass within the SM. However, many questions in the field of HEP are still unanswered, which motivates precision measurements of the SM as well as searches for physics beyond the Standard Model.

The measurements performed at the LHC are based on extensive, multi-stage analysis pipelines. Several algorithms are deployed along those pipelines, and many of those algorithms are nowadays either complemented or even fully replaced by machine learning (ML) techniques. So far, the corresponding ML models are usually developed and trained independently.

Over the last few years, the concept of foundation models [26] has emerged in other fields like natural language processing and computer vision. Foundation models are pre-trained on large datasets using a pre-training task. After pre-training, a foundation model can be fine-tuned either on another task or on the same task but with another dataset. While the pre-training of a foundation model can be time- and resource consuming, this pre-training can both drastically improve downstream performance and substantially reduce the required research and development (R&D) for models designed for those downstream tasks. Given the extensive use of ML in HEP, this paradigm is a promising way forward for HEP, and needs to be explored.

This thesis presents multiple studies in the context of foundation models for particle physics. Specifically, the methods developed and studied in this thesis focus on particle jets, which are sprays of particles created in a collision event. Particle jets are crucial objects at hadron colliders like the LHC, which makes them an optimal testbed for the development of foundation models in HEP. This poses two questions: i) what kind of data should be used for pre-training, and ii) what should be the pre-training task in the context of jet physics? The answers to those two questions are entangled. A supervised pre-training task for example requires external labels, like information about the particle that initiated a particle jet. In general, simulated particle physics data that includes such external labels is of high fidelity, which makes such supervised pre-training a legitimate option. At the same time, such external labels are not accessible in measured data from a collider experiment, which in the case of particle jets only detects the jet constituents themselves. This makes unsupervised and self-supervised pre-training tasks (which do not require external labels) attractive for particle physics, as collider experiments record huge amounts of data, which could then be used directly for pre-training. Industrial foundation models like large language models (LLMs) or models operating on image data are typically pre-trained on large, unlabelled datasets. In the case of natural language, the model learns to predict the next element in a

sequence of tokens, also referred to as next token prediction (NTP). Vision foundation models on the other hand rely on methods like masked modeling or self-distillation. Inspired by such pre-training methods, the HEP community has explored similar techniques for particle jets.

Five research contributions are presented in this thesis. The first contribution [1] explores the JetClass dataset – a public, large-scale R&D dataset – for generative modeling of particle jets. The model developed in that study is the first generative model that can generate not only the kinematic features of the jet constituents, but also the particle-ID and the trajectory displacement information. Next, the concept of NTP is studied as a pre-training task for jet foundation models [2]. NTP is the underlying concept of large language models, and is also used during the pre-training phase of those models. Here, this formalism is applied to particle jets, by pre-training a model to predict the next particle, which results in a generative model that can generate a jet as a sequences of particles autoregressively. The pre-trained model is then fine-tuned to another task, jet classification, where the model takes a whole jet as input and infers the type of particle that initiated the jet. Since NTP-based pre-training does not require any external labels, this pre-training can in principle be performed on measured data directly, where truth information of the underlying process that initiated the jet is not available. The question of whether pre-training with NTP on measured data can yield an advantage in a subsequent downstream task is addressed in the third contribution [3] of this thesis. In that study, a foundation model is pre-trained on open data from the CMS experiment at CERN, and subsequently fine-tuned to the JetClass dataset. In the fourth contribution [4], the pre-training paradigm introduced in Ref. [2] is revised, and two substantial improvements to the original setup are introduced. While the original NTP-based pre-training in Ref. [2] uses a discretized representation of the individual particles for both input and training targets, this study proposes a hybrid setup that combines the prediction of discrete token-IDs via NTP with continuous-valued vectors as model inputs. Secondly, the pre-training task itself is enhanced by combining next token prediction with masked token prediction. The fifth and last contribution [5] focuses on a comparison of multiple pre-training methods, providing a comprehensive analysis of different pre-training techniques with and without external labels.

The remaining part of this thesis is structured as follows. Chapter 2 provides a brief overview of the Standard Model and its limitations, followed by an introduction to collider physics. Next, the fundamental machine learning concepts relevant for this thesis are presented in chapter 3. Based on that, chapter 4 describes the concept of foundation models both in general and specifically for jet physics. Chapters 5 to 9 summarize the research contributions of this cumulative thesis. Finally, conclusions and an outlook are presented in chapter 10. The full-length articles of the individual research contributions can be found in the *Publications* chapter at very end of the document.

Part I

Theory and Methodology

2 Particle Physics

Particle physics, also referred to as *HEP*, studies the elementary particles of our universe and how those particles interact with each other. Elementary particles are the smallest, most fundamental entities humanity has observed so far. Those particles are smaller than 10^{-18} m and are often described as *point-like*. This chapter provides an overview of the theoretical background and the experimental context in which the research of this thesis is performed. Unless further specified, this chapter is based on Refs. [27–30]. Section 2.1 provides a short introduction to the Standard Model (SM) of particle physics, a theory that has been tested in particle physics experiments for decades and has held up impressively. This is followed by a brief discussion of the shortcomings of the SM in section 2.2. Section 2.3 then introduces fundamental concepts used to study elementary particles at particle collider experiments.

2.1 The Standard Model of Particle Physics

2.1.1 Elementary Particles and Fundamental Interactions

An overview of the particles included in the SM is shown in Figure 2.1. The particle content of the SM is split into multiple groups based on the properties of the different particles. On the highest level the particles are split into *fermions* and *bosons*. Fermions are the constituents that make up the matter that surrounds us. Bosons on the other hand are the mediators of the fundamental interactions described by the SM, which are the electromagnetic interaction, the weak interaction and the strong interaction. Interactions via the gravitational force are not described by the SM. The different particles are characterized by their mass and their quantum numbers, some of which like the particle spin and the electric charge are shown in the illustration in Figure 2.1.

All fermions carry a spin of $1/2$ and are divided into two main groups: leptons and quarks. Quarks are distinctly different from leptons in that they carry *color charge*. Both leptons and quarks come in three *generations*. This means that for each lepton and quark of the first generation, namely the up and down quarks (u and d), the electron (e) and the electron neutrino (ν_e), there are two additional versions with the same properties except for their mass. The higher-generation leptons are called muon (μ) and tau (τ), together with their corresponding neutrinos ν_μ and ν_τ , and the higher-generation quarks are called the charm (c), strange (s), top (t) and bottom (b). However, fermions of the second and third generation are unstable and eventually decay into first-generation fermions, which is why only first generation fermions form stable matter. Lastly, for each fermion there exists an *anti-fermion* which has the same mass as its fermion partner but opposite-sign charges like the electric charge.

The different gauge bosons – all of which carry a spin of 1 – correspond to the different interactions happening between particles. The bosons couple to particles that carry the corresponding charge of a certain interaction. The electromagnetic force, which is for instance responsible for binding electrons to atomic nuclei and thus the formation of atoms, is described in the SM through interactions via photons. Only electrically charged particles participate in electromagnetic interactions. This means that the electron, the muon, the tau, which all have the same electric charge as the elementary charge $e = 1.602\,176\,634 \times 10^{-19}$ C [32], and all quarks, with electric charge values of $-1/3e$ and $2/3e$, can interact electromagnetically. The weak force is mediated by three different particles: the electrically neutral Z boson and the

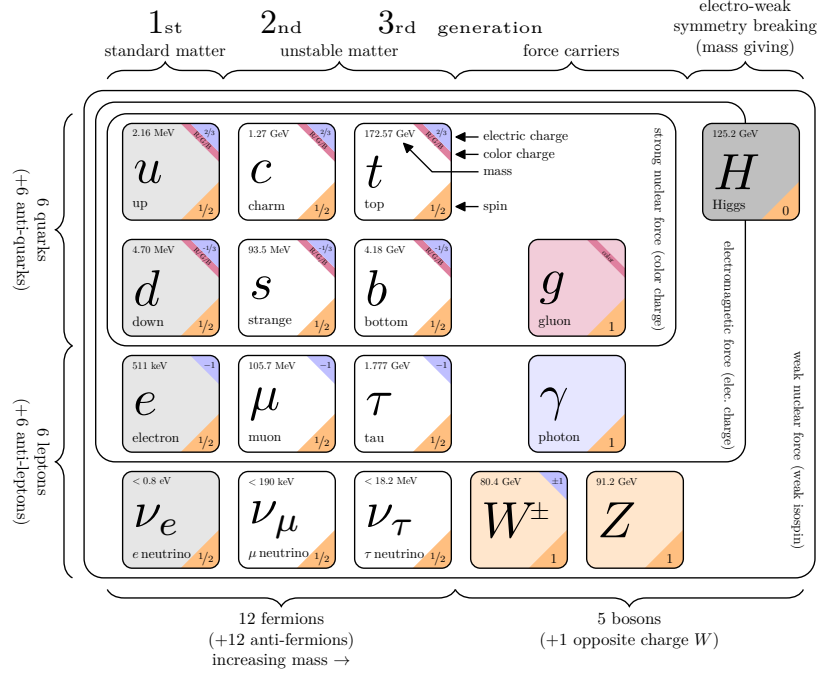


Figure 2.1: The particles of the Standard Model of Particle Physics. Figure adapted from Ref. [31] with updated mass values from Ref. [32].

two charged $W^\pm$ bosons, with the corresponding charge being the weak isospin. The strong force is a fundamental interaction in which all particles with color charge engage, with the gluon being the corresponding mediator particle. In contrast to e.g. electric charge, which can only take positive or negative values, this charge can take three different values, often denoted as green, blue and red, as well as the corresponding *anti* colors. Among the fermions, only quarks carry color charge. Additionally, gluons carry color charge themselves, which means that gluons can not only couple to quarks, but also to other gluons, which leads to gluon-gluon interactions. Another important property of the strong force is how its strength changes as a function of the energy scale of an interaction. At very high energies the strong force is in fact weak, which means that quarks behave approximately like free particles. This phenomenon is called *asymptotic freedom* [33, 34]. For low energies or large distances though, the potential energy between a quark-antiquark pair increases linearly. As a result, quarks can not exist as isolated particles and are instead only observed in the form of color-neutral bound states composed of multiple quarks, known as *hadrons*. This effect is known as *color confinement*.

Lastly, the Higgs boson is the only particle in the SM with spin 0. This particle emerges as a result of the *Higgs mechanism*, a theoretical framework for how massive particles in the SM acquire their mass.

2.1.2 The Standard Model as a Quantum Field Theory

At its core the SM is a Quantum Field Theory (QFT), combining the laws of special relativity and quantum mechanics to describe the particles introduced previously and their interactions. This section provides an overview of the overall concepts behind the QFT formalism of the SM. For an in-depth introduction and discussion, the reader is referred to one of many excellent textbooks about QFT, e.g. the previously mentioned Ref. [27–30]. Natural units are used from this point on throughout the whole thesis, meaning that the speed of

light c and the reduced Planck constant $\hbar$ are set to one. Furthermore, in this chapter, the Einstein summation convention is used for indices.

In classical mechanics, the Lagrangian of a system is given by

$$L(q_i, \dot{q}_i) = T(q_i, \dot{q}_i) - V(q_i), \quad (2.1)$$

where T is the kinetic energy and V is the potential energy. The q_i represent the generalized coordinates of the system (e.g. Cartesian coordinates, polar coordinates, ...) and $\dot{q}_i$ the corresponding time derivative. Given the Lagrangian, a system will take the trajectory that minimizes the action S , which is defined as the integral of the Lagrangian along a certain trajectory. Based on that fundamental law, the equations of motion can be obtained from the *Euler-Lagrange equation*

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) = \frac{\partial L}{\partial q_i}, \quad (2.2)$$

where the index runs over all spatial components.

QFT extends the concepts of Lagrangian mechanics to *particle fields*, offering a description that respects both quantum mechanics and special relativity. In contrast to classical mechanics, the time component has to be treated on par with spatial components. The Lagrangian in QFT is a density and does not depend on generalized coordinates q_i , but instead on fields $\psi_i(t, x, y, z)$, which depend on both space and time. The Lagrangian of a free spin $1/2$ particle, e.g. an electron that does not interact with any external fields, is given by the Dirac Lagrangian

$$\mathcal{L}_{\text{Dirac}} = i\bar{\psi}\gamma^\mu\partial_\mu\psi - m\bar{\psi}\psi, \quad (2.3)$$

where ψ is the particle's field (a so-called *Dirac spinor*), $\bar{\psi} = \psi^\dagger\gamma^0$ with $\psi^\dagger$ being the Hermitian adjoint of ψ , m is the particle mass and γ^μ are the gamma matrices. As in classical mechanics, the equations of motion can be obtained from the Euler-Lagrange equation, which in the case of QFT has the form

$$\partial_\mu \left(\frac{\partial \mathcal{L}}{\partial(\partial_\mu\psi_i)} \right) = \frac{\partial \mathcal{L}}{\partial\psi_i}, \quad (2.4)$$

where $\partial_\mu = (\partial_t, \partial_x, \partial_y, \partial_z)$ represents the gradient with respect to both time and the spatial coordinates.

Gauge Theories

Gauge invariance is another essential concept used in the QFT formalism of the SM. That is that the Lagrangian is invariant, meaning that it leads to the same measurable physical quantities, under a certain change of gauge. This change of gauge can be *global* or *local*. A global change could be e.g. to modify a field ψ_i as $\psi \rightarrow e^{i\theta}\psi$, meaning its phase is changed everywhere the same way in the form of a global phase shift. The field $\bar{\psi}$ then transforms as $\bar{\psi} \rightarrow (e^{i\theta}\psi)^\dagger\gamma^0 = e^{-i\theta}\bar{\psi}$. As a result, the factors $e^{i\theta}$ and $e^{-i\theta}$ cancel and the Lagrangian remains unchanged. The corresponding local change on the other hand depends on the spacetime vector x_μ , i.e. the field is modified as $\psi \rightarrow e^{i\theta(x_\mu)}\psi$. It is common to reparametrize $\theta(x_\mu)$ as $-q\lambda(x_\mu)$, and consequently the local transformation has the form $\psi \rightarrow e^{-iq\lambda(x_\mu)}\psi$. This choice is more convenient down the road as q can be associated with

the charge of the interaction. Transforming the spinor accordingly gives rise to an additional term in the Lagrangian from Equation 2.3:

$$\mathcal{L}_{\text{Dirac}} \rightarrow \mathcal{L}_{\text{Dirac}} + q(\bar{\psi}\gamma^\mu\psi)\partial_\mu\lambda \quad (2.5)$$

To achieve local gauge invariance, the *gauge term* $-q(\bar{\psi}\gamma^\mu\psi)A_\mu$ is added to the Lagrangian, with a vector field A_μ that transforms under the above-mentioned gauge transformation as

$$A_\mu \rightarrow A_\mu + \partial_\mu\lambda(x_\mu). \quad (2.6)$$

With those definitions, the gauge term cancels the additional term that emerges from the Dirac Lagrangian in Equation 2.5. By also adding the kinetic term that describes the dynamics of a vector field A_μ corresponding to a massless spin-1 particle, the whole Lagrangian becomes

$$\mathcal{L} = \underbrace{i\bar{\psi}\gamma^\mu\partial_\mu\psi - m\bar{\psi}\psi}_{\mathcal{L}_{\text{Dirac}}} - \underbrace{\frac{1}{4}F^{\mu\nu}F_{\mu\nu}}_{\text{kin. term of } A_\mu} - \underbrace{q(\bar{\psi}\gamma^\mu\psi)A_\mu}_{\text{interaction term}}, \quad (2.7)$$

with $F^{\mu\nu} = \partial^\mu A^\nu - \partial^\nu A^\mu$. Assuming a massive spin-1 particle here would require us to add a term $\frac{1}{2}m_A^2 A^\nu A_\nu$ to the Lagrangian as well. This term is not invariant under the transformation from Equation 2.6 and would consequently break the local gauge invariance again. Thus, the notion of a massless gauge boson is a rather natural choice as the goal here is to achieve local gauge invariance. As indicated in Equation 2.7, the individual terms of this new Lagrangian can be associated with the kinetic and mass terms of the spin-1/2 particle $\mathcal{L}_{\text{Dirac}}$, the kinetic term of the spin-1 particle and the interaction term which connects the field ψ with the field A_μ . The factor q in the interaction term now represents the *coupling constant* for this interaction.

This interaction is in fact the electromagnetic interaction, as introduced in Quantum Electrodynamics (QED) [35] with ψ corresponding to an electrically charged fermion and A_μ being the photon. Overall, this shows that requiring local gauge invariance in the form of local phase transformations $\psi \rightarrow e^{i\theta(x_\mu)}\psi$ leads to an elegant formulation of how electrically charged fermions interact via photons. All possible transformations of that type build the group U(1) of unitary 1×1 matrices, which is why this invariance is called U(1) invariance.

A more general way to extend a global gauge invariance to a local one is to introduce the *covariant derivative* D_μ and replace the derivative ∂_μ in the Lagrangian with D_μ . In the case of QED the covariant derivative is given by

$$D_\mu = \partial_\mu + iqA_\mu, \quad (2.8)$$

and the gauge invariant Lagrangian can be written as

$$\mathcal{L} = i\bar{\psi}\gamma^\mu D_\mu\psi - m\bar{\psi}\psi - \frac{1}{4}F^{\mu\nu}F_{\mu\nu}. \quad (2.9)$$

However, the derivation of the U(1)-invariant Lagrangian is a special case as the U(1) group is *Abelian*, with an Abelian group having the property that two elements of the group commute. For the non-Abelian (non-commutative) SU(N) groups the general representation of the gauge transformation becomes $\psi \rightarrow e^{i\theta^a T^a}\psi$, with T^a being the generators of the group and θ^a depending on the spacetime vector x_μ . The generators fulfill the commutation relation

$$[T^a, T^b] = if^{abc}T^c, \quad (2.10)$$

where the f^{abc} are the completely antisymmetric structure constants. For the $SU(2)$ group for example, the structure constants are given by the Levi-Civita symbol $f^{abc} = \epsilon_{abc}$, which is equal to one for even permutations of the indices (e.g. $f^{231} = 1$), minus one for odd permutations (e.g. $f^{132} = -1$), and zero if two indices are the same (e.g. $f^{112} = 0$). The covariant derivative and the transformation of the gauge fields A_μ^a are given by

$$D_\mu = \partial_\mu + igA_\mu^a T^a, \quad (2.11)$$

$$A_\mu^a \rightarrow A_\mu^a + \frac{1}{g} \partial_\mu \theta^a + f^{abc} A_\mu^b \theta^c, \quad (2.12)$$

with g being the coupling strength. With the field strength tensor $F_{\mu\nu}^a$, which is defined as

$$F_{\mu\nu}^a = \partial_\mu A_\nu^a - \partial_\nu A_\mu^a + gf^{abc} A_\mu^b A_\nu^c \quad (2.13)$$

the gauge-invariant Lagrangian becomes

$$\mathcal{L} = i\bar{\psi}\gamma^\mu D_\mu \psi - m\bar{\psi}\psi - \frac{1}{4} F_{\mu\nu}^a F_a^{\mu\nu}. \quad (2.14)$$

Using the formalism outlined up to here, the full Lagrangian of the SM that respects a $SU(3)_C \times SU(2)_L \times U(1)_Y$ symmetry can be derived. In this description the weak and the electromagnetic interaction are unified into the *electroweak* interaction [36–38], and the strong interaction is described by Quantum Chromodynamics (QCD) [39]. The index C represents the color charge, the index L stands for *left-handed* fields, and Y denotes the *hypercharge*, which is defined as $Y = 2Q - I_3$ with Q being the electric charge and I_3 the third component of the weak isospin. Left-handed fields transform as doublets under the $SU(2)_L$ group, whereas right-handed fields transform as singlets. The left- and right-handed components ψ_L and ψ_R of a spinor field ψ are obtained with the corresponding projection operators as

$$\psi_L = \left(\frac{1 - \gamma^5}{2} \right) \psi \quad \text{and} \quad \psi_R = \left(\frac{1 + \gamma^5}{2} \right) \psi, \quad \gamma^5 = i\gamma^0\gamma^1\gamma^2\gamma^3. \quad (2.15)$$

While this allows for an accurate description of strong and electroweak interactions, additional considerations are needed to explain the masses of the $W^\pm$ and Z bosons since, as mentioned before, introducing mass terms for the gauge bosons breaks the gauge invariance. In addition to that, using ordinary mass terms of the form

$$m\bar{\psi}\psi = m(\bar{\psi}_R\psi_L + \bar{\psi}_L\psi_R) \quad (2.16)$$

for fermions, as e.g. used in Equation 2.3, would violate gauge invariance as the left- and right-handed components of the fermion fields transform differently under $SU(2)$.

The Higgs mechanism [20–25] solves these challenges via *spontaneous symmetry breaking*. This involves introducing a complex scalar field doublet Φ , referred to as the *Higgs field*, with the Lagrangian

$$\mathcal{L} = |D_\mu \Phi|^2 - \mathcal{V}(\Phi). \quad (2.17)$$

The potential is given by

$$\mathcal{V}(\Phi) = -\mu^2 \Phi^\dagger \Phi + \lambda (\Phi^\dagger \Phi)^2 \quad (2.18)$$

where $\mu^2 > 0$ and $\lambda > 0$. This potential has degenerate ground states at $\Phi^\dagger \Phi = \mu^2/2\lambda = v^2/2$ with $v = \sqrt{\mu^2/\lambda}$ being the vacuum expectation value (VEV) of the Higgs field. When the

$SU(2) \times U(1)$ symmetry is broken spontaneously, the system transitions to a ground state at v . Evaluating the term $|D_\mu \Phi|^2$ in Equation 2.17 at the VEV gives rise to the gauge boson mass terms [29]. Additionally, the Higgs field allows relating left- and right-handed fermion fields in a gauge-invariant way, that yields mass terms for the fermions. The masses of the fermions are related to the VEV via a coupling constant λ_f as

$$m_f = \frac{\lambda_f}{\sqrt{2}} v, \quad (2.19)$$

which is known as *Yukawa coupling* [40]. Finally, the Higgs boson shows up as a fluctuation of the Higgs field with mass $m_H = \sqrt{2\lambda} v$. This was experimentally verified in 2012 by two independent measurements by the CMS and ATLAS collaborations at CERN [18, 19].

Free Parameters The SM features a total of 19 free parameters, which means that the values of those parameters are not set by the theory, but instead have to be determined experimentally. Three of those parameters correspond to the coupling constants of the three interactions. The QCD vacuum angle is another parameter of the SM. The fermion masses make up nine parameters (neutrinos are treated as massless in the SM) and the Cabibbo–Kobayashi–Maskawa (CKM) matrix, which describes quark flavor mixing requires four parameters. Finally, the Higgs sector accounts for the last two parameters, one of them being the Higgs boson mass and the other the VEV.

Perturbation theory To test the Standard Model, physicists use *perturbation theory* to obtain approximations of experimentally observable quantities. These predictions calculate transition amplitudes from the initial state to a certain final state expanded as a series of individual terms. The perturbative solution is given as a power series of the coupling parameter $\alpha = \frac{g^2}{4\pi}$, with g being the coupling strength of the corresponding interaction. Those terms can be depicted as *Feynman diagrams*, which represent the incoming (initial state) particles of an interaction, the outgoing (final state) particles, as well as the interactions and intermediate particles involved in the transition process. Thus, depending on the desired precision, different sets of Feynman diagrams are considered in these calculations.

2.2 Beyond the Standard Model

As mentioned at the beginning of this chapter, the Standard Model is an immensely successful theoretical framework that withstood extensive experimental tests. Nonetheless, the SM does have a few shortcomings, motivating both precision measurements of the SM, and measurements that target physics *beyond the Standard Model* (BSM). Some of those shortcomings are briefly discussed in this section. For a more extensive review of both the theoretical and experimental side of BSM physics, the reader is referred to the excellent review provided in [32].

Grand Unified Theories The strength of the electromagnetic, weak and strong force depend on the energy scale of the corresponding interaction. At large energy scales, the strength of the different forces is expected to become comparable. *Grand unified theories* attempt to unify the three different forces of the SM into an even more fundamental force, from which the electromagnetic, weak and strong force can be derived. A first attempt for a grand unified theory was presented already back in 1974 [41], but the corresponding predictions were disproven experimentally.

Neutrino Masses The SM assumes that neutrinos are massless particles. Despite the fact that this is a valid approximation for many reactions, measurements of neutrino oscillations indicate that neutrinos have non-zero mass, and individual neutrino flavors occur as mixtures of mass eigenstates. A model that extends the SM to realize neutrino masses is the *seesaw mechanism* [42–46]. As of today, there are different variants of the seesaw mechanism. The simplest (*type I*) variant postulates the existence of extremely heavy right-handed neutrinos (the SM contains only left-handed neutrinos). As a result, one obtains both very heavy and lightweight mass eigenstates, which could be compatible with the tiny neutrino masses inferred experimentally.

Quantum Gravitation One of the most prominent limitations of the SM is the absence of gravitational interactions. Gravitation is accurately described by the theory of general relativity [47], though this theory is not a quantum theory and can thus not be combined directly with the QFT formalism of the SM. A consistent theory of quantum gravity with a dedicated mediator particle, often referred to as *graviton*, would allow to incorporate gravity in the same theoretical framework.

Dark Matter There are multiple experimentally observed phenomena in the context of astrophysics and cosmology that can not be explained based on the known matter particles of the SM. When the motion of stars within galaxies and galaxy clusters is calculated based on the laws of general relativity, the predicted dynamics do not agree with the observed dynamics when the masses of stars and galaxies are calculated based on visible matter [48–50]. To obtain correct predictions, significantly larger masses have to be assumed, hinting towards the existence of matter that humans are not able to detect. This form of matter is generally referred to as *dark matter*. Gravitational lensing observations [51] also indicate the existence of dark matter, as the mass of visible matter does not agree with the mass required for the observed lensing effects [52]. Dark matter candidate particles do not interact via the electromagnetic force. Many theories for dark matter were proposed so far with a few examples being *sterile neutrinos* [53], *axions* [54–57] or *weakly interacting massive particles (WIMPs)*.

Hierarchy Problem When considering quantum corrections to the Higgs boson mass, one finds that this leads to corrections δm^2 that grow quadratically with the energy cutoff scale Λ of the theory [58, 59], i.e. the energy scale up to which the theory is assumed to be valid. Given the fact that the Higgs boson mass was measured as $(125.2 \pm 0.1) \text{ GeV}$ [32], large values of Λ require precise fine-tuning of the uncorrected Higgs boson mass. This becomes especially challenging when considering energy scales $\Lambda \approx 1 \times 10^{16} \text{ GeV}$ of *grand unified theories*. Due to the large differences in scale, this is known as the *hierarchy problem*. A theoretical framework that solves this, but has not been verified experimentally despite extensive searches, is *Supersymmetry* [59].

Matter-Antimatter Asymmetry It is expected that matter and antimatter was produced in equal portions during the Big Bang, followed by matter-antimatter annihilation. Yet, today’s universe consists only of matter [60, 61]. This asymmetry is called *matter-antimatter asymmetry* or *baryon asymmetry*. There are many theoretical mechanisms that explain this asymmetry, with a few examples being theories that involve superheavy particles [62, 63], supersymmetric models [64], and baryon asymmetry generation at the electroweak scale [65, 66].

A range of experiments is deployed to probe the SM and to search for physics beyond the Standard Model. The research presented in this thesis was performed in the context of *particle collider* experiments.

2.3 Collider Physics

Quite a few elementary particles were discovered before the first notable collider experiments were built. The electron was discovered back in 1897 [67]. The photon gradually became to be considered an elementary particle starting with Einstein proposing that light is made of discrete quanta in 1905 [68]. Two physicists then discovered the muon in 1936 while studying cosmic radiation [69], and the electron neutrino was detected in 1956 [70], confirming the particle postulated to explain the apparent violation of conservation of energy in beta decay. A few years later, the muon neutrino was discovered in a fixed-target accelerator experiment at the Brookhaven National Laboratory (BNL) [71]. Deep inelastic scattering experiments (also called fixed-target experiments) at the SLAC National Accelerator Laboratory (SLAC) in the late 1960s revealed that protons are composite particles [72, 73], leading to the discovery of quarks. Yet, to explore the potential existence of heavy particles like the τ lepton, W boson or Z boson, higher energies were needed, with collider experiments being the first experiments to detect those particles [74, 75].

In particle collider experiments, particles are accelerated to velocities close to the speed of light, with two beams pointing in opposite directions. Those high energy particles are then brought to collision, inducing interactions of elementary particles at enormous energies. The interaction point is surrounded by a particle detector that allows to measure the particles created in the collision event – either directly or indirectly via their decay products. Over the last few decades, multiple collider facilities were built and operated, reaching for increasingly high energies and with that discovering the particles of the SM one after the other. An illustration of the history of particle collider experiments overlaid with the time of discovery of different elementary particles is shown in Figure 2.2. The different collider experiments are grouped by the type of colliding particles: lepton colliders, hadron colliders and lepton-hadron colliders. Which types of particles are used for the collision events depends on the physics program of the experiment:

- **Lepton colliders:** These colliders usually collide electrons and positrons. Since the initial state of the colliding particles is well-defined, these colliders yield clean and well-understood collisions. Thus, they are the preferred choice for precision measurements.
- **Hadron colliders:** Hadron colliders predominantly collide (anti-) protons and heavy ions. In the context of circular colliders, an important component to consider is the energy loss due to Synchrotron radiation. This energy loss scales with $\propto \frac{1}{m^4}$, m being the mass of the accelerated particle. As protons are roughly 2000 times heavier than electrons, hadron colliders allow for much higher collision energies than (circular) lepton colliders. This makes them the common choice when aiming for the discovery of new heavy particles. However, since hadrons are composite particles, the initial state of the colliding particles is not well-defined, which leads to messy collision events that are much harder to reconstruct than those at lepton colliders.
- **Lepton-hadron colliders:** This type of collider allows for precise measurements of the internal structure of hadrons, providing insights into QCD. The corresponding collision events are not as complex as the ones seen at hadron colliders, but also not as clean as those at lepton colliders.

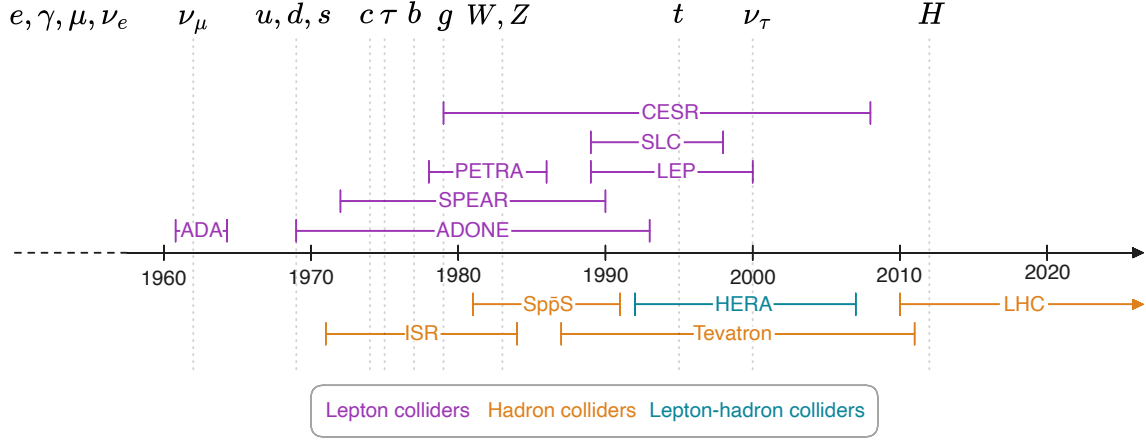


Figure 2.2: Operation times of a selection of particle colliders, as well as the year of discovery of all known elementary particles.

The first discovery of a new elementary particle at a collider experiment happened in 1974 with the discovery of the charm quark with the SPEAR collider at SLAC [76], though the same discovery was made simultaneously by a fixed-target experiment at BNL [77]. Next, evidence for the τ lepton was found with the SPEAR collider [74]. Over the following years, many elementary particles were discovered via the experiments shown in Figure 2.2. In 1995, the top quark discovery at the Tevatron collider [78, 79] marked the discovery of the heaviest elementary particle known so far. And lastly, with the discovery of the Higgs boson [18, 19] in 2012 at the LHC, the last missing piece of the SM was found, as it confirmed the Higgs mechanism.

2.3.1 The Large Hadron Collider

The Large Hadron Collider (LHC) [16, 17] is the highest-energy particle collider to date. It is located in Geneva, Switzerland, and is operated by the European Organization for Nuclear Research (CERN). The LHC is designed to accelerate and collide either protons or heavy ions. Four main experiments are deployed along the 27 km accelerator ring: ATLAS [14], CMS [15], LHCb [80] and ALICE [81]. The ATLAS and CMS experiments feature general-purpose particle detectors, and cover a wide range of physics analyses. The ALICE experiment is focused on heavy-ion collisions and the LHCb experiment is specifically designed for measurements of CP violation. The LHC was designed for proton-proton collisions at a center-of-mass energy of 14 TeV, with the so far maximum realized value being 13.6 TeV in Run III.

2.3.2 Particle Detectors

The studies in this thesis concern collision events at the LHC measured with either the CMS or the ATLAS detector. While each detector has its own special design choices, they both feature the same main components, which are briefly discussed in the following. A more detailed description can be found in Ref. [15] for the CMS detector and in Ref. [14] for the ATLAS detector.

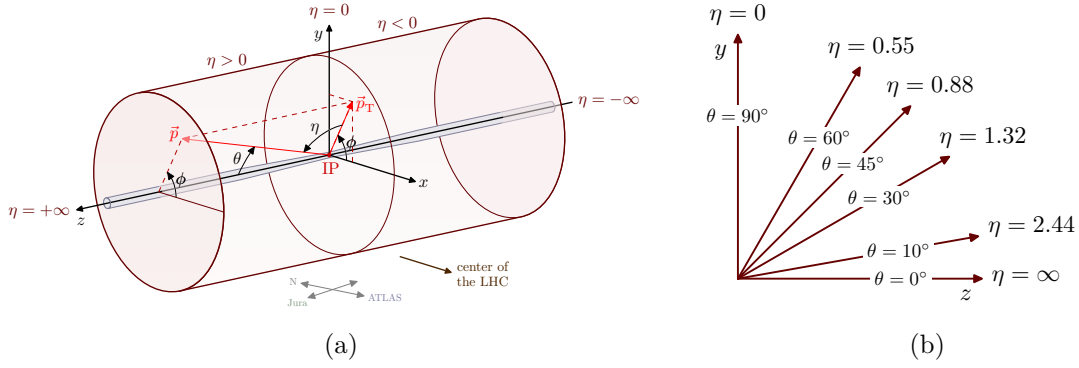


Figure 2.3: Illustration of the coordinate system used at the LHC: (a) illustration of a momentum vector $\vec{p}$ in the CMS coordinate system with its projection onto the transverse plane $\vec{p}_T$ and (b) illustration of pseudorapidity values for different values of the polar angle θ . Figures taken from Refs. [82, 83].

Coordinate System and Kinematics

The coordinate system used by ATLAS and CMS is shown in Figure 2.3. The origin of the coordinate system is located at the interaction point (IP), where the incoming particles collide. Directions in the xy -plane, also referred to as the transverse plane, are described by the azimuthal angle ϕ . The z -axis of the coordinate system points along the beam axis, and the polar angle θ is measured towards the z -axis. However, instead of using the polar angle directly, collider experiments predominantly use the *pseudorapidity* η , which is directly related to the polar angle via

$$\eta = -\ln \left[\tan \left(\frac{\theta}{2} \right) \right]. \quad (2.20)$$

The spatial distance between two vectors is generally quantified with

$$\Delta R = \sqrt{(\Delta\phi)^2 + (\Delta\eta)^2}, \quad (2.21)$$

where $\Delta\phi$ is the difference of the azimuthal angle of the two vectors. Pseudorapidity is a high-energy approximation ($E \gg m$) of rapidity y , which is defined as

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right). \quad (2.22)$$

Experiments at the LHC usually use pseudorapidity to describe the *forward* direction of a particle since the particle production at the LHC is approximately flat in η , and differences in pseudorapidity $\Delta\eta$ are invariant with respect to Lorentz boosts along the z -axis (in the high-energy approximation). This is particularly useful at hadron colliders as the colliding particles, i.e. the quarks and gluons inside the protons, generally carry unequal fractions of the proton's momentum, and thus the center-of-mass frame of the collision is boosted along the z -axis.

An important quantity in collider experiments like ATLAS and CMS is the transverse momentum $\vec{p}_T$, which is the projection of a momentum vector onto the transverse plane:

$$\vec{p}_T = \begin{bmatrix} p_x \\ p_y \end{bmatrix}, \quad p_T = \sqrt{p_x^2 + p_y^2}. \quad (2.23)$$

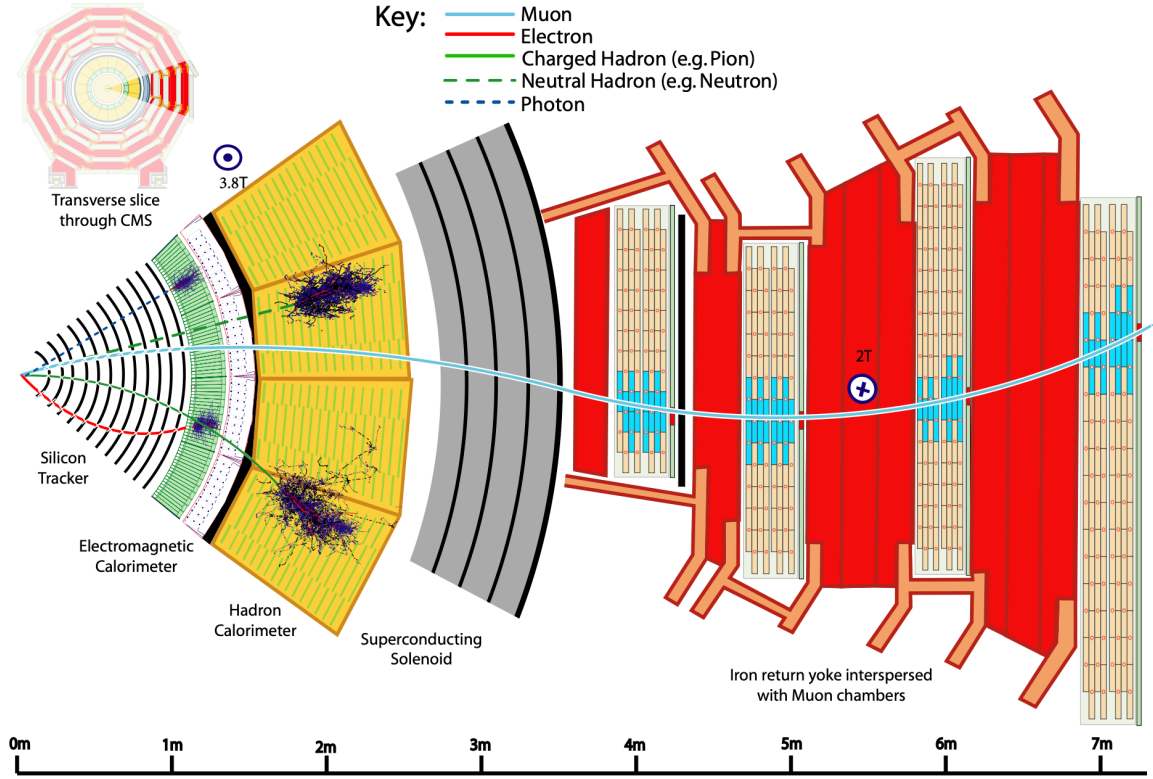


Figure 2.4: Transverse slice through the CMS detector with example trajectories and particle showers for different types of particles. Figure taken from Ref. [84].

Since the transverse momentum of the incoming protons is zero, the transverse momenta of all outgoing particles have to sum to zero as well. Thus, observing a non-vanishing sum of the transverse momenta corresponds to one or multiple particles not being detected. This is commonly referred to as *missing transverse momentum* (the 2D vector representation) or *missing transverse energy* (the magnitude of the missing transverse momentum), and is essential for measurements that target for example dark matter candidates that do not interact with the detector.

While the typical four-momentum representation used in special relativity is (E, p_x, p_y, p_z) , the same information is given by the representation (p_T, η, ϕ, m) with

$$m = \sqrt{E^2 - p_x^2 - p_y^2 - p_z^2} \quad (2.24)$$

being the invariant mass of a particle. This alternative representation is more convenient for most use-cases in experimental particle physics, and mostly used throughout this thesis.

Tracking Detector

The innermost detector component, i.e. the one closest to the interaction point, of both ATLAS and CMS is the tracking detector. This detector component is designed for the reconstruction of the trajectories of charged particles, also referred to as *tracks*. Reconstructed tracks are in turn used to reconstruct *vertices*, which are grouped into *primary vertices* and *secondary vertices*. Primary vertices correspond to the point where particles from the two beams collide. Secondary vertices are the points where unstable particles produced at the primary vertex decay. The tracking detector consists of very small silicon detector components

and is immersed in a strong magnetic field, which bends the trajectories of charged particles. Based on the curvature of the trajectories, the momenta of the corresponding particles can be inferred. This is visualized in Figure 2.4, where charged particles are shown with curved trajectories in the transverse plane of the detector. Neutral particles do not leave a track in the tracking detector and are thus indicated with dashed lines in the illustration.

Calorimeter

Calorimeters measure the energy of particles by absorbing their energy. Both ATLAS and CMS feature two main calorimeter components: the electromagnetic calorimeter (ECAL) to measure the energy of photons and electrons and the hadronic calorimeter (HCAL) to measure the energy of hadrons. Those calorimeters are implemented as *sampling calorimeters* or *homogeneous calorimeters*. Sampling calorimeters consist of alternating layers of absorber (passive) material and active material. Incoming particles interact with the absorber material, leading to the creation of new, lower-energy particles. This process happens repeatedly, leading to *particle showers*. The layers comprising the active material measure the energy of this shower. In homogeneous calorimeters, the same material acts as both the absorber and the active material. The purple clusters in Figure 2.4 illustrate the particle showers of different types of particles. Photons and electrons are stopped within the ECAL, whereas hadrons shower primarily in the HCAL.

Muon Detector System

The outermost detector component is the muon detector system, dedicated to the measurement of the momenta of muons. It surrounds the components mentioned previously, i.e. the tracking detector and the calorimeter system, since muons are highly penetrating particles that can pass through those components without being stopped. Like the tracking detector, the muon detector system is immersed in a magnetic field that bends the trajectories of electrically charged particles.

Trigger System

The LHC operates at a collision frequency of 40 MHz, corresponding to proton bunches colliding every 25 ns. With a file-size of $\mathcal{O}(1 \text{ MB})$ per event [85, 86], storing all 40 million events per second would require the ability to write 40 terabytes per second to disk. This is not feasible, which is why only a small fraction of the recorded events are stored on disk for subsequent analysis. Thus, a sophisticated trigger system is needed to decide which events should be stored and which should be discarded. In both ATLAS and CMS, the event rate is first reduced from 40 MHz to 100 kHz by a hardware-based *level-1 trigger*. The remaining events are then further processed by a software-based *high-level trigger* that reduces the event rate to $\sim 7 \text{ kHz}$ in CMS [87] and $\sim 3 \text{ kHz}$ in ATLAS [88].

Particle Reconstruction

Based on the readouts from the various detector components, particles are *reconstructed* with dedicated algorithms. In the CMS experiment, particles are reconstructed with the *particle-flow (PF)* algorithm [84], which combines information from all different sub-detector systems simultaneously. It correlates different *PF elements*, which are tracks, vertices, and energy clusters measured in the calorimeters via a *linking algorithm*. This results in *PF blocks* of elements that are either linked directly or indirectly through a common linked element. The PF blocks are then used for the particle identification and particle reconstruction.

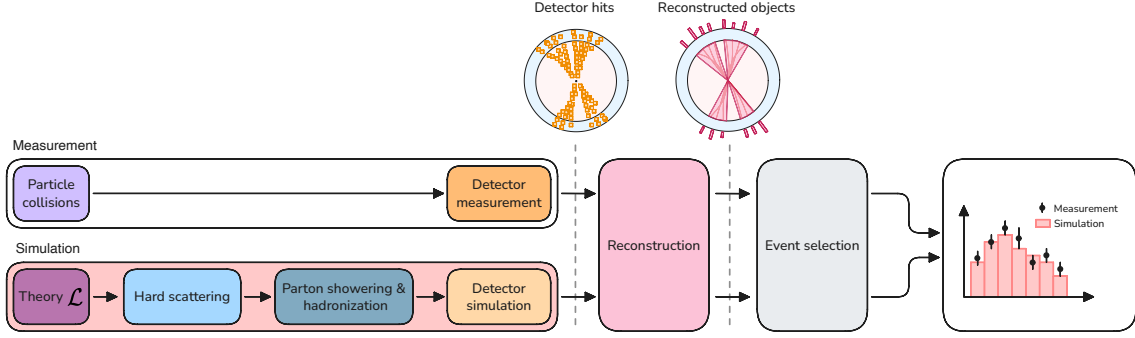


Figure 2.5: High-level sketch of the typical data analysis workflow at the LHC. Simulated data is obtained from Monte Carlo (MC) simulation, divided into multiple stages. Measured and simulated detector hits are reconstructed with dedicated algorithms to yield reconstructed objects corresponding to information of individual particles or collections of particles.

The algorithm starts with the identification and reconstruction of muon candidates. Next, electrons, as well as energetic, isolated photons are reconstructed. After this step, a cross-identification of charged hadrons, neutral hadrons and photons follows. Finally, once all blocks have been processed by the PF algorithm, the reconstructed event is post-processed. This post-processing focuses on adjusting events with artificially large missing transverse momentum due to a misidentified or misreconstructed high- p_T muon. The reconstructed particles are then called *PF candidates*, and are simply called *particles* in the following.

The LHC Data Analysis Workflow

On a high level, most analyses performed at the LHC follow the workflow illustrated in Figure 2.5. The end goal is to compare simulated events obtained from theoretical predictions to the events measured with a detector, revealing if those predictions provide an accurate description of nature’s laws of physics. This essentially means that there are two streams of data: the measured data and the simulated data. It should be noted that the usual naming convention in HEP is to call the *measured data* simply “data” and never use the term “data” for simulated events. The naming convention used in this thesis is less strict, and both measured and simulated entities are referred to as “data”, with a corresponding prefix if clarification is needed. The individual steps shown in Figure 2.5 are briefly described in the following. It should be noted though that each of the steps discussed below is immensely involved, and vast amounts of time and computational resources are needed to accomplish each of these steps for a given analysis at the LHC.

Measurement The actual particle collision events are measured directly with the detector, resulting in detector hits. The detector signals are then reconstructed as described in section 2.3.2.

Simulation To obtain the corresponding counterpart as predicted by theory, a multi-stage simulation, mostly based on Monte Carlo (MC) methods, is used. An illustration of a simulated $pp \rightarrow t\bar{t}$ event is shown in Figure 2.6, with the individual simulation stages being briefly discussed in this paragraph. An extensive explanation of the individual simulation steps can be found in [89]. The simulation starts with the hard scattering of the incoming partons, with each parton belonging to one of the two incoming hadrons. While PYTHIA [89–

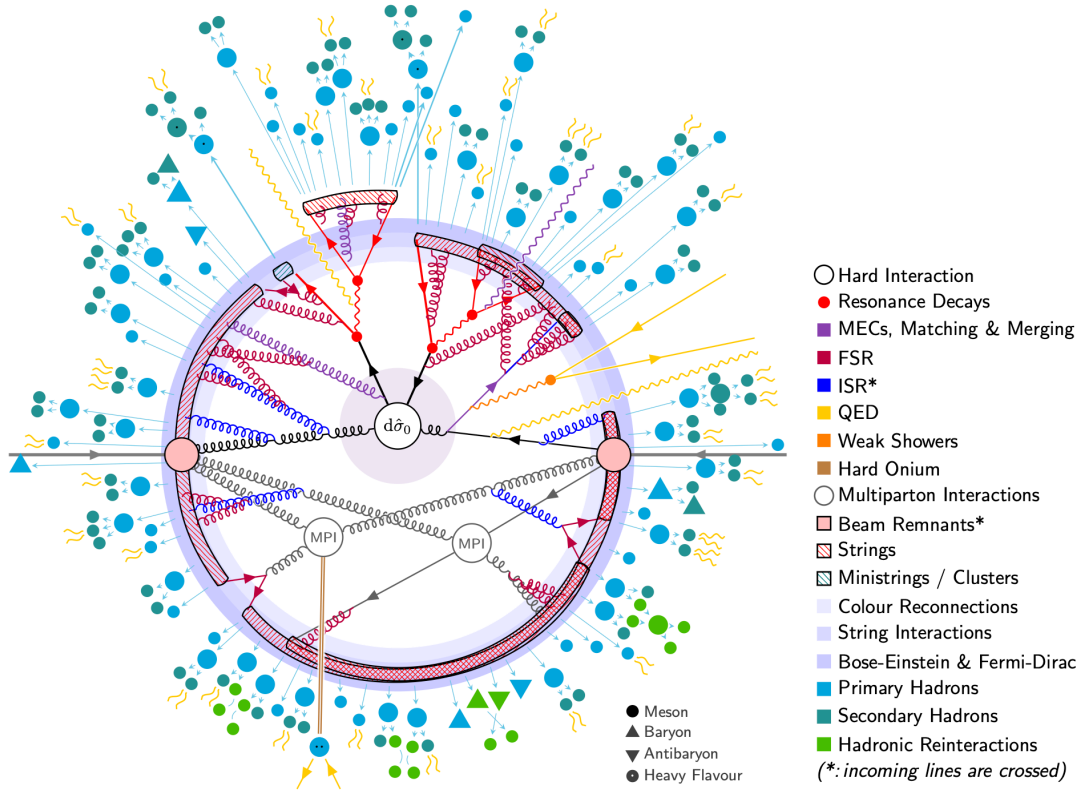


Figure 2.6: Schematic illustration of a $pp \rightarrow t\bar{t}$ event as modelled by PYTHIA [89]. Figure taken from Ref. [89].

91] can simulate the hard scattering of a few simple processes, it is more common to simulate the hard scattering with other packages like HERWIG [92, 93], SHERPA [94–96], POWHEG [97–101] or MADGRAPH [102] and then import the result of the hard scattering simulation into PYTHIA for the subsequent simulation steps. The hard scattering simulation is performed using perturbative calculations as introduced in section 2.1.2. The hard scattering produces short-lived particles such as top quarks in this example. Fixed-order radiative corrections can be added via matrix-element corrections, matching and merging strategies. Next, the *parton showering* is simulated, which is split into initial state radiation (ISR) and final state radiation (FSR). ISR refers to the radiation of additional particles from the incoming particles, while FSR corresponds to such radiation from the final state particles (i.e. the ones produced in the hard scattering). These radiation and pair-production processes occur at high energies, which means that perturbative QCD calculations yield accurate simulations. However, throughout the parton showering process, the energy scale becomes lower and with that the strength of the strong interaction increases. Next, the *hadronization* process is simulated, which describes the formation of color-neutral bound states. As perturbative calculations do not hold up at the corresponding energy scale, the hadronization process has to be simulated by phenomenological models instead. Such phenomenological models are e.g. the cluster model [103] or the Lund string model [104], with the latter being used in the PYTHIA [89–91] simulator. As colored particles are separated from each other, the potential energy stored in the associated strings increases. Once the potential energy becomes large enough, strings break up and new quark-antiquark pairs are created which results in the fragmentation of color strings into hadrons. Unstable hadrons decay further, until only

stable particles remain. This yields a list of four-vectors of particles, which are then used as input for the *detector simulation* (not shown in Figure 2.6), with the standard program for detector simulation being GEANT4 [105, 106]. This program simulates how the particles interact with the different detector components, resulting in a data representation of the same format as the measured data. After the detector simulation, the simulated data is reconstructed with the same algorithms as the measured data.

Event Selection and Data Analysis Analysis-specific selection criteria are applied to both data streams. Finally, the selected, reconstructed events obtained from measurement and simulation respectively are analyzed, revealing insights into the validity of the theoretical predictions.

Fast Simulation

Simulated particle physics data is of incredible fidelity. However, the computational resources required to simulate large amounts of such high-quality data are massive. Less accurate, but much more efficient *fast simulation* programs like DELPHES [107] are therefore used for many exploratory studies, where the *full simulation* with GEANT4 [105, 106] would be computationally too expensive. This compromise is well justified for a wide range of R&D work, with many public benchmark datasets like the top tagging reference dataset [108, 109] or the JETCLASS dataset [110, 111] that are also used in this thesis. However, for actual physics analyses performed by collaborations like ATLAS or CMS, full simulation is needed to obtain the most reliable results possible. In view of the upcoming upgrade to the High-Luminosity Large Hadron Collider (HiLumi-LHC), this poses a huge challenge. The HiLumi-LHC is a major upgrade to the LHC that will increase the integrated luminosity, which is a measure for the recorded data size, by a factor of ten [112]. To benefit from this massive increase of collected data, a similar amount of simulated data is needed. Yet, with currently available methods, the experiments will not be able to keep up with this increased demand for full simulation and thus alternative options have to be explored [113]. A promising advancement in this direction is the development of artificial intelligence (AI)-based fast detector simulation [114].

2.3.3 Hadronic Jets

Since quarks and gluons can not exist as free particles, they can not be measured directly as opposed to e.g. electrons. Instead, they lead to particle showers with subsequent hadronization as described in section 2.3.2. The corresponding objects that can be observed experimentally are called *particle jets*. A particle jet is usually described as a *collimated spray of particles*, i.e. a collection of reconstructed particles that point in a similar direction in the detector. Measuring and reconstructing particle jets thereby provides a way of inferring properties of the original quark or gluon. Particle jets are crucial objects for almost every analysis performed at the LHC. Thus, the HEP community has put a lot of effort into studying those objects in high detail. As the research presented in this thesis focuses exclusively on studies with particle jets, a few core concepts of jet physics are introduced in the following. Detailed introductions to jet physics can be found in [115, 116].

Jet Clustering Algorithms

A jet is constructed by clustering and combining multiple reconstructed particles. This is illustrated in Figure 2.7, which shows a schematic representation of a collision event with

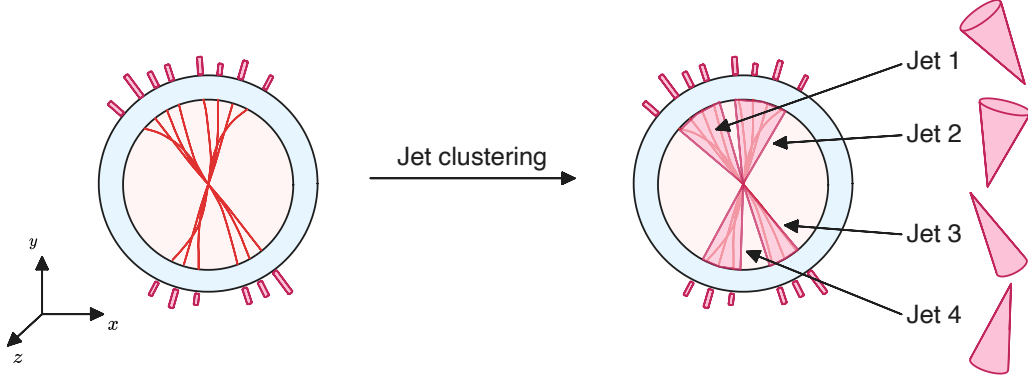


Figure 2.7: Illustration of how the reconstructed particles of an event are clustered into four jets.

the corresponding particles being clustered into four jets. The sketch shows a rather clear separation of four sprays of particles that allows for an intuitive assignment of which particles should be grouped together. However, real collider events are much busier in that they contain many more tracks, and thus sophisticated clustering algorithms are needed. A first definition of so-called *cone jets* was already presented in 1977 [117]. Since then, many jet algorithms have been proposed and studied with the aim of ending up with a standardized definition of particle jets [118, 119]. The algorithms most commonly used in modern collider physics are discussed in the following. For a more comprehensive review of the history of particle jet clustering algorithms the reader is referred to [119–122].

There are two main categories of jet clustering algorithms: *cone algorithms* and *sequential recombination algorithms*. Cone algorithms follow a top-down approach in that they assign jets to regions in the detector where a lot of energy flows. Sequential recombination algorithms on the other hand follow a bottom-up approach and iteratively combine particles based on a *distance* measure. Experiments at the LHC usually use sequential recombination algorithms for jet clustering, which follow the procedure outlined in algorithm 1. This generalized structure was originally presented in [123], which unified previously published algorithms [124–127]. It should be noted that jets are usually illustrated as cones even though not all clustering algorithms return conical jet shapes.

input : A set of particles $\{p_1, \dots, p_n\}$ and a resolution parameter R

output: A set of jets $\{j_1, \dots, j_m\}$

while there are *pseudojets* left **do**

 Calculate distances d_{ij} between all pairs (p_i, p_j) ;

 Calculate distances d_{iB} ;

 Find smallest distance within $\{d_{ij}\} \cup \{d_{iB}\}$;

if the smallest distance is a d_{ij} **then**

 Recombine p_i and p_j ;

else

 Remove p_i and call it a jet;

end

end

Algorithm 1: Generalized structure of sequential recombination algorithms for jet clustering.

The input for the algorithm is a set $\{p_1, \dots, p_n\}$ of particles p_i . The p_i are in the following referred to as *pseudojets*, with a pseudojet representing either the four-momentum of a particle from the original input set or the four-momentum corresponding to a recombination of one or more particles. In each iteration the algorithm calculates the distance matrix d_{ij} , with d_{ij} being the distance between the entities p_i and p_j . Additionally, the distances d_{iB} between the pseudojets and the beam B are calculated. Next, the smallest value within those distances is determined. If the smallest distance corresponds to a distance d_{ij} , then the pseudojets p_i and p_j are recombined, usually via a summation of the four-momenta, reducing the number of elements in the set of pseudojets by one. Otherwise, i.e. if the smallest distance belongs to a distance d_{iB} , the corresponding pseudojet p_i is called a jet and is removed from the set. This procedure is then repeated with the updated set of pseudojets until no pseudojets are left.

The difference between the most commonly used jet clustering algorithms is the *definition of the distances* d_{ij} and d_{iB} . In the generalized form [123], these are defined as

$$d_{ij} = \min(k_{ti}^{2p}, k_{tj}^{2p}) \frac{\Delta_{ij}^2}{R^2}, \quad (2.25)$$

$$d_{iB} = k_{ti}^{2p}, \quad (2.26)$$

with $\Delta_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$ representing the distance in the rapidity-azimuth plane, k_{ti} being the transverse momentum of pseudojet i , and R being a resolution parameter, often called *jet radius*. The parameter p allows controlling the relation between the energy scale and the geometric scale throughout the clustering process. Using $p = -1$ results in the anti- k_t [123] algorithm, which tends to build the clusters around high- p_T particles. Choosing $p = 1$ on the other hand yields the k_t algorithm [124, 125], which clusters nearby low- p_T particles first. The Cambridge/Aachen algorithm [126, 127], which corresponds to $p = 0$ ignores the particle's p_T , and clusters solely based on the angular distance Δ_{ij} . The most commonly used algorithm is the anti- k_t algorithm, i.e. $p = -1$, which leads to cone-like jets despite *not* being a cone algorithm. Lastly, the clustering is generally performed using FASTJET [128, 129], which provides an efficient implementation of the here described sequential recombination algorithms.

Clustered jets can then be used as objects in an analysis. However, in order to obtain finite and meaningful theoretical predictions of jet observables, certain requirements have to be fulfilled. To this end, jet clustering algorithms used nowadays in experiments at the LHC are required to be infrared and collinear (IRC) safe:

- **Infrared safety:** the jet algorithm returns the same jets if an infinitely soft (i.e. low-energy) particle is added.
- **Collinear safety:** the jet does not change if a particle splits into two collinear particles, i.e. two particles moving in the same direction.

The algorithms discussed here, i.e. the k_t , anti- k_t and the Cambridge/Aachen algorithm, all satisfy those requirements.

Jet Substructure

If a heavy initial particle like a top quark is boosted, i.e. carries a large momentum, then the corresponding decay products are so collimated that they are reconstructed as a single, large-radius jet. It is important for LHC analyses to be able to distinguish different signal-type jets, e.g. the ones originating from top quarks, W/Z bosons, or Higgs bosons from

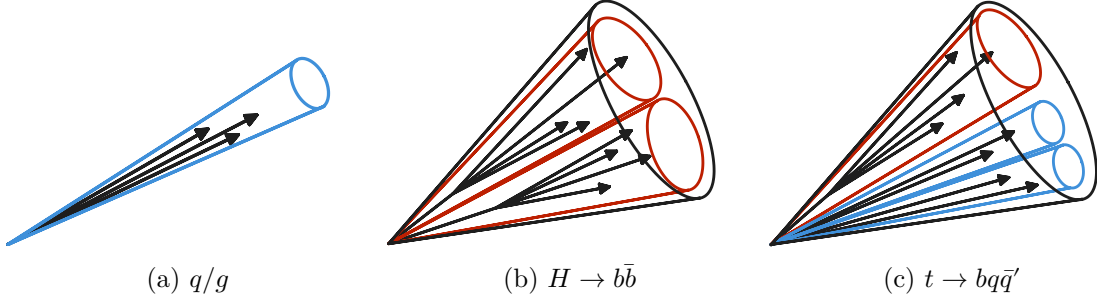


Figure 2.8: Illustration of different jet types when splitting the whole, large-radius jet (black) into subjets, corresponding to the involved b -quark (red) and light quarks / gluons (blue). q/g jets (a) generally do not feature subjets, whereas $H \rightarrow b\bar{b}$ jets (b) can be often split into two clear subjets and $t \rightarrow bq\bar{q}'$ jets (c) tend to split into three subjets.

background jets, i.e. q/g jets that originate from a light quark or a gluon. Thus, the concept of exploiting the underlying jet *substructure* was introduced in the context of boosted jet identification [130–140]. A wide range of methods exists in the space of jet substructure. In the following the basic concept of the *jet mass* and the concept of *subjettiness* are discussed in more detail.

By definition, a jet represents an object that combines multiple particles. Thus, by combining the four-momenta of the individual jet constituents¹, a jet initiated by for example a $t \rightarrow bq\bar{q}'$ decay is expected to result in a four-momentum closely related to the original top quark. One can then calculate the invariant mass of the jet, which should follow a distribution centered around the top quark mass. Conversely, for jets originating from an $H \rightarrow b\bar{b}$ decay, one expects a jet mass distribution centered around the mass of the Higgs boson. But aside from features of the final jet four-momenta, one would expect those two types of jets to look somewhat different, especially in terms of how the decay products can be divided into *subjets*. This is illustrated in Figure 2.8, which shows sketches of three different types of jets: q/g , $H \rightarrow b\bar{b}$, and $t \rightarrow bq\bar{q}'$. A q/g jet (Figure 2.8a) is expected to show a spray of particles centered around the jet axis, i.e. the four-momentum that is assigned to the jet after clustering. For an $H \rightarrow b\bar{b}$ jet (Figure 2.8b) one can envision that the two quarks from the $b\bar{b}$ -pair are separated by a certain angle. Each quark then hadronizes and forms its own subjet, shown in the Figure as two smaller-radius red jets within the overall jet cone. And finally for the $t \rightarrow bq\bar{q}'$ jet (Figure 2.8c) it is plausible that the jet constituents can be grouped into three subjets, one belonging to the b -quark and the other two to the light quarks $q\bar{q}'$, as those light quarks arise from the hadronic W decay in $t \rightarrow bW \rightarrow bq\bar{q}'$. A way of quantifying how well a large-radius jet can be divided into N subjets, is the concept of N -subjettiness [140]. This observable is usually defined as

$$\tau_N = \frac{1}{d_0} \sum_k p_{T,k} \min \{ \Delta_{1,k}, \Delta_{2,k}, \dots, \Delta_{N,k} \}, \quad (2.27)$$

with the sum running over all jet constituents, and with $p_{T,k}$ being their transverse momenta. $\Delta_{i,k}$ is the distance between the k -th constituent and the i -th subjet axis in the rapidity-azimuth plane, and d_0 is a normalization factor.

¹The terms *particles* and *jet constituents* are used interchangeably.

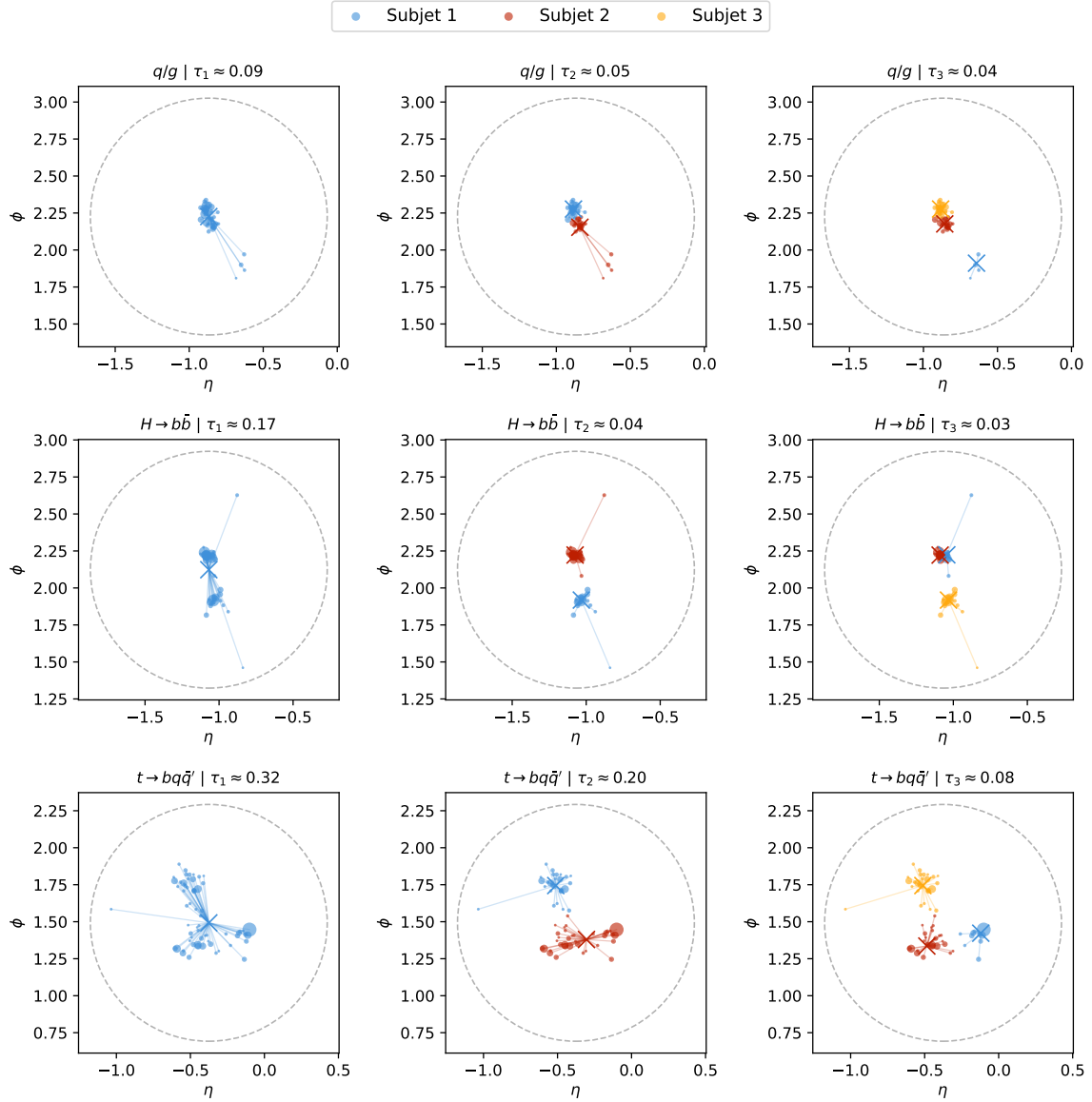


Figure 2.9: Three individual jet examples and how they re-cluster into one, two or three exclusive subjects with the k_t algorithm. Each row shows an example of a certain jet type: q/g (top), $H \rightarrow b\bar{b}$ (middle), and $t \rightarrow bq\bar{q}'$ (bottom). For each example, the re-clustering into the different numbers of subjects is shown, with the corresponding jet axes indicated with a cross. The individual particles are shown as scatter points in the η - ϕ plane, with the size of each dot scaled with the p_T of the particle, and connected via a line to the axis of the subject they are assigned to. The jet examples are taken from the JETCLASS dataset.

This normalization factor is defined as

$$d_0 = \sum_k p_{T,k} R_0, \quad (2.28)$$

with R_0 being the jet radius of the original jet clustering. The choice of the subject axis is a parameter of the subjeettiness. A common choice is to use the subject axes obtained by re-clustering the particles into N subjects with the exclusive k_t algorithm. Even more informative than the bare value of a subjeettiness τ_N are the *subjeettiness ratios*. For example,

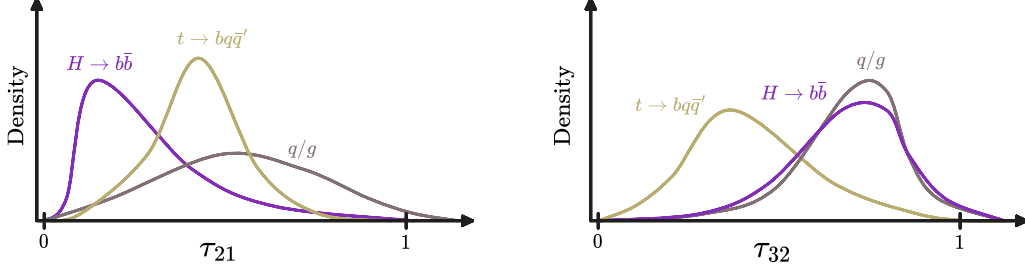


Figure 2.10: Qualitative illustration of the subjeettiness ratio distributions for the three jet types q/g , $H \rightarrow b\bar{b}$ and $t \rightarrow bq\bar{q}'$.

the subjeettiness ratio $\tau_{32} := \tau_3/\tau_2$ can be seen as a quantification of the question “*do those particles align much better into three subjets compared to two subjets?*”, and the same goes for $\tau_{21} := \tau_2/\tau_1$ with two subjets vs. one subjet. Representative examples of simulated jets from the public JETCLASS dataset [110, 111] are shown in Figure 2.9. The JETCLASS dataset contains boosted anti- k_t jets with a radius of $R = 0.8$, illustrated as a dashed circle. Each row shows a single jet, with the first row showing a q/g jet, the second row an $H \rightarrow b\bar{b}$ jet, and the third row a $t \rightarrow bq\bar{q}'$ jet. The different columns show the corresponding re-clustering into one, two, or three exclusive subjets with the k_t algorithm. For each jet and re-clustering into N subjets, the corresponding subjeettiness τ_N is calculated and shown in the title of each panel.

When looking at the q/g jet in the top row of Figure 2.9, one can see that most constituents are quite close to the jet axis, and the re-clustering into two or three subjets does not separate the jet into clear subjets. This is reflected in the corresponding subjeettiness ratios $\tau_{21} \approx 0.6$ and $\tau_{32} \approx 0.8$. The $H \rightarrow b\bar{b}$ jet in the middle row of Figure 2.9 shows a *2-prong* structure, which becomes apparent when re-clustering into two subjets. As a result, the two-subjeettiness is much smaller than the one-subjeettiness with a ratio $\tau_{21} \approx 0.24$. At the same time, the three-subjeettiness is not much smaller than the two-subjeettiness, with $\tau_{32} \approx 0.75$ and thus this jet is presumably more consistent with two subjets instead of three subjets. Lastly, the $t \rightarrow bq\bar{q}'$ jet in the bottom row of Figure 2.9 shows an example which is well compatible with a re-clustering into three subjets, also called a *3-prong* jet. In terms of the subjeettiness ratios, this especially shows up in the small value of $\tau_{32} \approx 0.4$.

While these are just individual examples, they motivate the main idea of N -subjeettiness and their ratios. Statistically, jets of the three different types discussed so far follow distributions qualitatively shown in Figure 2.10. As can be seen in those distributions, the subjeettiness ratio τ_{21} can be used to separate 2-prong jets (like $H \rightarrow b\bar{b}$ jets) from q/g jets, and the ratio τ_{32} can be used to separate 3-prong jets like $t \rightarrow bq\bar{q}'$ from q/g jets. Additionally, subjeettiness ratios can also be used to evaluate the quality of simulated jets. These are common observables when evaluating machine learning based simulation explored in this thesis, comparing the corresponding simulated jets to jets obtained with the simulation chain described in section 2.3.2.

3 Machine Learning

Machine learning (ML) is a subset of artificial intelligence (AI) and describes the process of fitting mathematical models to observed data through an optimization process referred to as *training*. Today, machine learning (ML) methods are essential for a wide range of applications. In particle physics, ML-based methods are nowadays involved in all stages of the HEP analysis workflow outlined in section 2.3.2. Reviews about the profound impact of ML in particle physics can be found in Refs. [141–148].

This chapter provides a brief introduction to the topic of ML and the relevant concepts for the research presented in this thesis. First, an overview of the different subfields of ML and the associated loss functions is provided in section 3.1, followed by an introduction to how ML models are trained in section 3.2. Based on that, section 3.3 covers the basics of neural networks, the most popular type of machine learning model in modern applications. Next, the subfields of generative machine learning and masked modeling are discussed in section 3.4 and section 3.5, respectively. Lastly, the concept of tokenization is introduced in section 3.6. Unless specified further, this chapter is largely based on Refs. [149, 150].

3.1 Overview

Generally, a machine learning model f_{θ} takes some input x and transforms it into some output $f_{\theta}(x)$, with the subscript θ representing the *trainable parameters* of the model. What the input x and the output $f_{\theta}(x)$ represent, in which data spaces they live, and how they are related varies between different tasks. Quite often the true, analytical relationship between the input and the output is not known or the relationship between input and output is simply intractable. In such cases, ML provides a powerful way of modeling the corresponding relationship. Another motivation to use ML methods is to simplify the modeling of a certain relationship. This is useful when a more traditional method is maybe accurate, but computationally very expensive. In that situation, ML models can potentially learn to achieve the same performance while at the same time being more computationally efficient. Such models are known as *surrogate models*.

3.1.1 Loss Functions

ML models are trained by minimizing a *loss function* $\mathcal{L}$. For now, it is assumed that there is a mechanism that enables training of a model f_{θ} for a certain task as long as the problem can be phrased as an optimization problem on a loss function. How this training process is realized is covered in section 3.2. The loss function quantifies how well a model is doing at the task it is trained for and a lower loss value corresponds to better performance. The choice of loss function heavily depends on the task at hand, as well as on the availability of *labels* in the dataset. The latter is commonly used to split the field of machine learning into the two subfields *supervised learning* and *unsupervised learning*, discussed later. This might suggest that labels are used during the training process whenever available. Yet, it should be noted that the availability of labels alone does not dictate whether those should be included in the training process.

For classification tasks, the common choice for the loss function is the *cross entropy loss*. When evaluated on N data points, and with K possible classes, the cross entropy loss is

given by

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_i^N \sum_k^K y_{i,k} \log(f_{\boldsymbol{\theta}}(x_i)_k), \quad (3.1)$$

where y_i is a K -dimensional one-hot encoded vector, that is $y_{i,k} = 1$ if the i -th data point belongs to class k , and $y_{i,k} = 0$ otherwise.

In regression, where the target is an M -dimensional continuous vector $y_i \in \mathbb{R}^M$, the model is usually trained with the mean squared error (MSE) loss, which is given by

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_i^N \|y_i - f_{\boldsymbol{\theta}}(x_i)\|^2, \quad (3.2)$$

or with the mean absolute error (MAE)

$$\mathcal{L}_{\text{MAE}} = \frac{1}{N} \sum_i^N |y_i - f_{\boldsymbol{\theta}}(x_i)|. \quad (3.3)$$

3.1.2 Supervised Learning

Supervised learning concerns the case where the training dataset consists of N labeled tuples $\{(x_i, y_i)\}$ with $i = 1, \dots, N$, and the model $f_{\boldsymbol{\theta}}$ is trained to predict the correct label y_i given the input x_i . Common supervised learning tasks are:

- **Classification**, where y_i is discrete and corresponds to a class label. The model predicts the probability of x_i belonging to a certain class. As the true labels are given in the dataset, the model is optimized to match those as closely as possible.
- **Regression**, where y_i is continuous and corresponds to an M -dimensional target vector in $\mathbb{R}^M$. The model predicts the target vector directly.

3.1.3 Unsupervised Learning

Unsupervised learning concerns cases where the model training only involves the data points $\{x_i\}$, and the model learns the internal structure of the data. Quite often this involves learning the probability distribution $p(x)$. Typical tasks included in this category are:

- **Generative modeling**: learn $p(x)$ such that the model can be used to generate new samples $x \sim p(x)$.
- **Density estimation**: learn the distribution $p(x)$ and use it to *evaluate the probability* of a sample afterwards. Can be seen as asking “*How likely is this observation x_i under the model $f_{\boldsymbol{\theta}}$?*”.
- **Clustering**: learn a mapping that divides the data space into K representative classes (without knowing beforehand what those classes are).

A subset of unsupervised learning in the sense of not using external labels is the field of *self-supervised learning*. Today, self-supervised learning is typically treated as a separate category, yet in this thesis it is treated as a subset of unsupervised learning, since it does not require external labels. In self-supervised learning tasks, labels are created from the input itself. This means that in terms of its implementation, the task itself can be for example a classification or regression task, but the corresponding targets are obtained from the input rather than using external labels. Common self-supervised tasks are:

- **Masked prediction:** in this framework, the input is *masked*, which means that part of the input is removed or replaced with some fill values. The target is to predict the masked part x_{masked} given the information x_{survived} that survived the masking procedure by learning the conditional probability $p(x_{\text{masked}}|x_{\text{survived}})$.
- **Next token prediction:** this method is commonly used for modeling sequences, with the most prominent application being natural language models. Based on an input sequence, the model predicts the next element (often referred to as a “token”) in that sequence. The task is to learn the conditional probability $p(x_{S+1}|x_1, \dots, x_S)$, with the index representing the index within the sequence and S being the sequence length of the input. Note that this is also a *generative modeling* task listed above in the unsupervised learning category.
- **Autoencoding:** an *autoencoder* compresses the input x into a lower-dimensional *latent space* and then reconstructs this latent representation back to the original space. The autoencoder is trained to minimize a reconstruction error $d(x, f_{\theta}(x))$, with $d(\cdot, \cdot)$ being a distance measure. After training, the autoencoder can for example be used for anomaly detection by applying it to new examples that are flagged as anomalous if the reconstructed representation differs significantly from the input.

The loss functions used for unsupervised and self-supervised tasks vary, and depend on the task and model architecture, where the architecture refers to the design choices of how the input is transformed into the output of the model. A generative model for instance can be designed in various ways, with different methods using distinct loss functions. For example, the next token prediction task, which is discussed in more detail in section 3.4.2, can be phrased as a classification task, and is thus implemented with the cross entropy loss. At the same time, other generative modeling paradigms involve a regression task, as for example the flow matching paradigm described in section 3.4.1.

3.2 Model Optimization

ML models are trained by minimizing the loss function $\mathcal{L}$ as a function of the model parameters $\theta \in \mathbb{R}^P$, where P is the number of free parameters of the model. Thus, the goal is to find parameters $\hat{\theta}$ such that

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} [\mathcal{L}(\theta)] , \quad (3.4)$$

where the loss $\mathcal{L}$ is calculated on the whole dataset $\mathcal{D}$, with $\mathcal{D} = \{(x_i, y_i)\}$ in the supervised learning case and $\mathcal{D} = \{x_i\}$ in the unsupervised case. In practice, this is usually achieved with iterative updates of the model parameters θ , which are also referred to as *weights*. There are various methods to achieve this iterative training procedure. Those methods, which are essentially sets of rules on how model parameters are updated each iteration, are called *optimizers*.

Gradient Descent The standard way of training modern ML algorithms is to use some form of *gradient descent*. This method follows the idea of adjusting the parameters θ via the corresponding gradient $\nabla_{\theta} \mathcal{L}$ of the loss function. For a model with P parameters, this

gradient is given by

$$\nabla_{\boldsymbol{\theta}} \mathcal{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \frac{\partial \mathcal{L}}{\partial \theta_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \theta_P} \end{bmatrix}. \quad (3.5)$$

In the purest form of gradient descent, the parameters are then updated each iteration via

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}, \quad (3.6)$$

with α being a positive scalar value called the *learning rate*. An update step in gradient descent is also referred to as a *gradient step*. At the beginning of the training, the parameters $\boldsymbol{\theta}$ are initialized randomly. As the update steps point in the opposite direction of the gradient, the loss decreases with each update, with the learning rate determining the size of the update steps. The learning rate is a so-called *hyperparameter*, which is a parameter that influences the training process, but is chosen before the training is performed and is not affected by the gradient descent. Using a large learning rate usually leads to a faster minimization of the loss in the early stage of the training, but can lead to overshooting once the loss minimum is approached, and can even lead to divergence of the loss. A small learning rate value on the other hand leads to precise, incremental adjustments of the parameters. However, this also means that the number of update steps required to reach the minimum increases. In addition to that, the algorithm can get stuck in a local, non-optimal minimum of the loss. The learning rate can also be adjusted during training using a so-called *learning rate scheduler*, which defines how the learning rate evolves as a function of the update step count.

Stochastic Gradient Descent A common variant of gradient descent is *stochastic gradient descent*. Instead of calculating the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ on the whole dataset $\mathcal{D}$, the gradient steps are calculated on a subset of the dataset, called a *minibatch* or simply *batch*. This leads to noisy gradient steps, which might seem suboptimal at first, but in fact stochastic gradient descent (SGD) has advantages over the plain gradient descent. On one hand it is simply more practical in terms of computational efficiency compared to computing the loss value on the entire dataset for every parameter update. In addition to that, the noisy (batch-wise) gradient steps can in principle escape local minima.

Momentum Modern optimizers modify stochastic gradient descent by adding some form of *momentum* to the update rule. This means that the update at iteration $t + 1$ is a combination of the gradients calculated from the current batch and the gradients calculated from the batch in the previous step (t). The update rule becomes

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1 - \beta) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t) \quad (3.7)$$

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \cdot \mathbf{m}_{t+1}, \quad (3.8)$$

where $\mathbf{m}_t$ is the momentum, α is the learning rate, and $\beta \in [0, 1)$ is a hyperparameter that controls to which extent the current update step is influenced by previous gradients. If consecutive gradients align, then this leads to a larger effective learning rate. Conversely, if the gradients of multiple steps point into different directions, those tend to cancel out as $\mathbf{m}_t$ is a vector, and thus the effective learning rate is reduced.

An issue of gradient descent with a fixed step size is that the corresponding update leads to large steps for parameters with large gradients and to small update steps for parameters associated with small gradients. This can result in too large steps for parameters with large gradients, and at the same time avoiding exploration of the parameter space in regions where the current state shows small gradients. The ADAM [151] optimizer addresses these issues by adaptively changing the learning rate based on the gradient $\nabla_{\theta} \mathcal{L}$ and the element-wise squared gradients. To this end, the ADAM update rule first defines the two quantities $\mathbf{m}_t$ and $\mathbf{v}_t$ as

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1 - \beta) \cdot \nabla_{\theta} \mathcal{L}(\theta_t) \quad (3.9)$$

$$\mathbf{v}_{t+1} \leftarrow \gamma \cdot \mathbf{v}_t + (1 - \gamma) \cdot (\nabla_{\theta} \mathcal{L}(\theta_t))^2, \quad (3.10)$$

with $\beta \in [0, 1)$ and $\gamma \in [0, 1)$ being momentum coefficients. To avoid very small effective learning rates at the beginning of the training, these statistics are further modified to

$$\tilde{\mathbf{m}}_{t+1} \leftarrow \frac{\mathbf{m}_{t+1}}{1 - \beta^{t+1}} \quad (3.11)$$

$$\tilde{\mathbf{v}}_{t+1} \leftarrow \frac{\mathbf{v}_{t+1}}{1 - \gamma^{t+1}}, \quad (3.12)$$

where the superscript in the above equations is an exponent, not an index of the iteration. As β and γ are smaller than one, the effect of this re-scaling vanishes as the iteration counter t becomes large. The final update rule is then given by

$$\theta_{t+1} \leftarrow \theta_t - \alpha \cdot \frac{\tilde{\mathbf{m}}_{t+1}}{\sqrt{\tilde{\mathbf{v}}_{t+1}} + \epsilon}, \quad (3.13)$$

where the square root in the denominator is applied element-wise and ϵ is a small, scalar value to avoid division by zero.

Further variants of the ADAM optimizer are for example ADAMW [152] and RADAM [153]. ADAMW modifies ADAM by adding *decoupled weight decay*, which helps in keeping the norm of the model weights from blowing up while at the same time not influencing the adaptive learning rate. RADAM (Rectified ADAM) addresses the issue of large variance of the adaptive learning rate during the early stage of the training.

Lookahead Optimizer The Lookahead [154] optimizer adds another level to the update rule, by introducing the concept of fast and slowly moving weights. This means that the optimization loop is nested: the outer loop represents the slow weights updates, and the inner loop the fast weights updates. Slow weights are denoted as ϕ_t , and fast weights as $\theta_{t,i}$, where the index t represents the iteration of the outer loop, and the index i represents the iteration of the inner loop. Within each iteration t of the outer loop, k optimization steps are performed with the fast weights $\theta_{t,i}$. At the beginning of this inner loop, the fast weights are set to the slow weights obtained from the last slow weights update, i.e.

$$\theta_{t,0} \leftarrow \phi_{t-1}. \quad (3.14)$$

Next, the k optimization steps are performed according to the update rules of the inner optimizer (e.g. ADAM), returning $\theta_{t,k}$. The slow weights are then updated by an update in the direction of $\theta_{t,k} - \phi_{t-1}$, scaled by the slow weights step size α :

$$\phi_t \leftarrow \phi_{t-1} + \alpha (\theta_{t,k} - \phi_{t-1}), \quad (3.15)$$

where $\alpha \in (0, 1]$ is a hyperparameter. The inner optimizer has its own learning rate and potentially further hyperparameters. For $\alpha = 1$ the weights trajectory obtained from the Lookahead optimizer corresponds to the trajectory returned from the inner optimizer without any adjustments. Choosing $\alpha < 1$ means that only part of the (by the inner optimizer) proposed update step is performed on the slow weights. Looking at the weights trajectory, this essentially means that the weights are first adjusted for k steps, but then in the final step (when the slow weights are updated) one moves “back” by a certain amount. The concept of the Lookahead optimizer is in general compatible with any choice for the inner optimizer. A popular choice is to use the RADAM optimizer as the inner optimizer, which is known as RANGER [155].

Generalization When training ML models, it is important to ensure that the model *generalizes* to unseen data. That is, the model can not only make useful predictions when the input data was part of the training dataset, but also yields comparable performance on samples that were not used in the optimization process. A typical example of a model that *does not generalize* would be a classification model that performs very well on the training dataset, but leads to worse performance when applied to new data. The common way of avoiding this is to split the dataset into three parts: training, validation, and testing. The training set is the data the model is trained on, the validation set is the one used to monitor the model’s generalization capabilities throughout the training and tune hyperparameters, and the test set is used for a final, unbiased evaluation of the model (after training). If a model does not generalize well, it is said to suffer from *overfitting* or *overtraining*. In practice, this can show up as a too large gap between the training loss and the validation loss. Different methods exist to prevent a model from overfitting, such as *regularization* techniques, for example L1 and L2 regularization that add explicit terms to the loss function, or methods like *early stopping* that stop the training if the performance on the validation set does not improve over a certain number of update steps.

3.3 Neural Networks

Modern machine learning models are usually built using *neural networks* (NNs), and the ML models used in HEP are no exception. This section provides a brief overview of neural networks in general, followed by an introduction to specific network architectures for point cloud data.

3.3.1 Multi-Layer Perceptron

There is a huge variety in how NNs are designed. However, most popular NN architectures involve the concept of *multi-layer perceptrons* (MLPs). An MLP is a function that transforms a D_{in} -dimensional input vector $\mathbf{x} \in \mathbb{R}^{D_{\text{in}}}$ into a D_{out} -dimensional output vector $\mathbf{y} \in \mathbb{R}^{D_{\text{out}}}$. As the name suggests, an MLP consists of multiple *layers*. Each layer transforms its n -dimensional input $\mathbf{u} \in \mathbb{R}^n$ into a new m -dimensional representation $\mathbf{v} \in \mathbb{R}^m$ via a calculation of the form

$$\mathbf{v} = \phi(\mathbf{W}\mathbf{u} + \mathbf{b}) , \quad (3.16)$$

where $\mathbf{W} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are trainable parameters, commonly referred to as *weight matrix* and *biases*, and $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is a non-linear *activation function* that is applied element-wise. MLPs are composed of an input layer, (multiple) hidden layers, and an output layer.

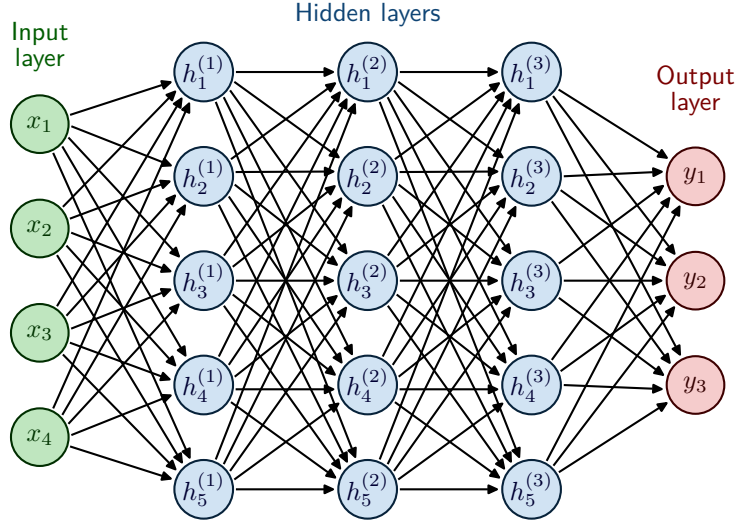


Figure 3.1: Example of a multi-layer perceptron (MLP) with three hidden layers with five nodes each. Input nodes are labeled as x_i , output nodes as y_i and the nodes of the hidden layers are denoted as $h_i^{(\ell)}$ with ℓ being the number of the corresponding hidden layer. Figure adapted from Ref. [156].

The number of hidden layers, their dimensionality, and the activation function are hyperparameters that are chosen before training the model. An example of an MLP with three hidden layers is shown in Figure 3.1. In this case the *forward pass*, i.e. the calculation of the output vector given the input vector, involves four consecutive operations as in Equation 3.16. The network has input dimension $D_{\text{in}} = 4$, all hidden layers have dimension five, and the output dimension is $D_{\text{out}} = 3$. Thus, the corresponding weight matrices have the dimensionality $\mathbf{W}^1 \in \mathbb{R}^{5 \times 4}$, $\mathbf{W}^2 \in \mathbb{R}^{5 \times 5}$, $\mathbf{W}^3 \in \mathbb{R}^{5 \times 5}$, and $\mathbf{W}^4 \in \mathbb{R}^{3 \times 5}$, where the upper index refers to the index of the four operations as in Equation 3.16, and the respective biases have the dimensionality of the corresponding layer. Different activation functions are used in practice. For hidden layers, a common activation function is the Rectified Linear Unit (ReLU) function:

$$\text{ReLU}(x) := \max(0, x). \quad (3.17)$$

The activation function applied to the output layers is chosen based on the model's task. For instance in a regression task the output layer activation function can be simply the identity. For a multi-class classification task on the other hand, the common choice is the *Softmax* activation function, which normalizes the values of the output nodes of the neural network to sum to one:

$$y_i = \text{Softmax}(z_i) := \frac{\exp(z_i)}{\sum_{k=1}^K \exp(z_k)}. \quad (3.18)$$

$\mathbf{z} \in \mathbb{R}^K$ is referred to as the *logits* vector of a multi-class classification problem with K classes, and the post-activation output can be interpreted as a vector with each entry y_i corresponding to the probability of the input entity corresponding to class i .

To train neural networks with gradient descent as described in section 3.2, one needs to compute the gradients of the loss with respect to the network parameters. This is achieved with the *backpropagation* [157–160] algorithm, which involves both a forward and a backward pass. The forward pass yields the model output from which the loss can be calculated.

The backward pass then calculates how the loss would change with respect to each parameter, starting with the last layer. Backpropagation exploits the chain rule of calculus, to efficiently calculate the gradients for earlier layers by reusing calculations from the gradient computation of later layers, thus propagating the loss derivatives backwards through the neural network. The numerical training procedure is commonly implemented with computing libraries like PYTORCH [161].

3.3.2 Network Architectures for Point Clouds

Many data structures can be represented as a D -dimensional vector $\mathbf{x} \in \mathbb{R}^D$ when modified accordingly, and can then be used as the input for a neural network with an MLP architecture. However, there exist more sophisticated neural network architectures that take the overall structure of the input data into account. Those neural network architectures often combine fundamental building blocks such as MLPs and aggregation functions in a way that respects the symmetries of the input data to obtain more powerful network architectures.

Point clouds are an important data structure in particle physics, used to represent objects such as collections of detector hits or jet constituents. A point cloud is a set $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of vectors $\mathbf{x}_i \in \mathbb{R}^D$, where each vector represents a data point and the individual points do not have an inherent order. Thus, the two representations $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, and $\{\mathbf{x}_2, \mathbf{x}_1, \dots, \mathbf{x}_N\}$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are swapped via a permutation in the latter, represents the same point cloud. Different network architectures that respect this symmetry exist. Using a notation similar to Ref. [162], a function $f_{\theta}(\{\mathbf{x}_i\})$ that operates on a point cloud $\{\mathbf{x}_i\}$ is permutation *invariant*, if for any permutation of elements π

$$f_{\theta}(\pi(\{\mathbf{x}_i\})) = f_{\theta}(\{\mathbf{x}_i\}), \quad (3.19)$$

i.e. the output of the function does not change when the input is permuted. The function $f_{\theta}(\{\mathbf{x}_i\})$ is permutation *equivariant* if

$$f_{\theta}(\pi(\{\mathbf{x}_i\})) = \pi(f_{\theta}(\{\mathbf{x}_i\})), \quad (3.20)$$

which means that the final result is the same if the permutation is applied to the input before passing it to the function, or to the output of the function obtained from the non-permuted point cloud. In the case of *equivariance*, the function f_{θ} returns a point cloud of the same cardinality. The same concept applies to other symmetries and the corresponding transformations of the symmetry group.

A simple yet powerful permutation invariant architecture for sets is the concept of Deep Sets [162], where the forward pass is split into two networks ϕ and ρ :

$$f_{\theta}(\{\mathbf{x}_i\}) = \rho \left(\sum_i \phi(\mathbf{x}_i) \right) \quad (3.21)$$

The network ϕ is applied to each element individually, i.e. it is a point-wise neural network. The corresponding output set $\{\phi(\mathbf{x}_i)\}$ is summed over, and then passed to the second neural network ρ . The summation introduces permutation invariance, as the result of the sum is independent of the order of the elements. Thus, the final result is permutation invariant as well. In principle, other permutation invariant aggregation operations, such as averaging, min-pooling or max-pooling, can be used as well.

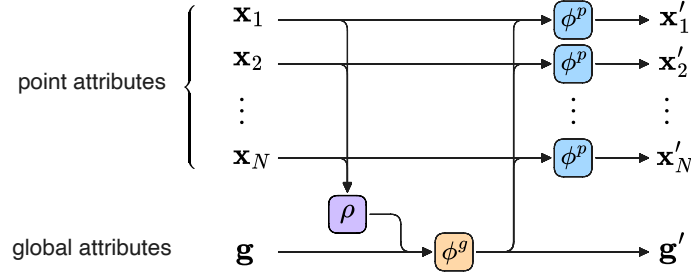


Figure 3.2: Forward pass through an EPiC layer. Figure adapted from Ref. [163].

EPiC Layers

Equivariant Point Cloud (EPiC) layers [163] are permutation equivariant neural network layers that can transform point clouds in a computationally efficient way. The forward pass through a single EPiC layer is visualized in Figure 3.2. The point cloud is described by the points $\mathbf{x}_i$ as well as a global vector $\mathbf{g}$, which captures information about the whole point cloud. The point cloud and the global vectors are then updated according to

$$\mathbf{g}' = \phi^g(\mathbf{g}, \rho(\{\mathbf{x}_i\})) , \quad (3.22)$$

$$\mathbf{x}'_i = \phi^p(\mathbf{g}', \mathbf{x}_i) , \quad (3.23)$$

where ρ is an aggregation function that calculates both the sum and the mean over the points, ϕ^g is an MLP that updates the global vector using the input global vector and the result from the aggregation, and ϕ^p is an MLP that updates the point-wise vectors using the updated global vector and the previous point vector as input. Multiple EPiC layers can be stacked to obtain a powerful, permutation equivariant network architecture.

Transformers

The Transformer [164] architecture is an extraordinarily successful neural network architecture. Transformers are based on the concept of *attention* [164] and were initially developed for natural language processing (NLP). While natural language has a clear ordering that must be respected when processed with a transformer, the attention mechanism is at its core permutation equivariant and can be used to process point cloud data [165, 166]. In the following, the attention mechanism is explained for point cloud data. Thus, it operates on sets $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with each $\mathbf{x}_i$ being a D -dimensional vector $\mathbf{x}_i \in \mathbb{R}^D$.

The first step in the attention calculation is to calculate so-called *queries*, *keys*, and *values*. In a matrix representation, the set $\{\mathbf{x}_i\}$ can be written as $\mathbf{X} \in \mathbb{R}^{N \times D}$. The queries, keys and values are denoted as matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. In *self-attention*, those matrices are calculated from the input $\mathbf{X}$:

$$\mathbf{Q} = \mathbf{X} \mathbf{W}^Q, \mathbf{K} = \mathbf{X} \mathbf{W}^K, \text{ and } \mathbf{V} = \mathbf{X} \mathbf{W}^V, \quad (3.24)$$

with projection matrices $\mathbf{W}^Q \in \mathbb{R}^{D \times d_q}$, $\mathbf{W}^K \in \mathbb{R}^{D \times d_k}$, and $\mathbf{W}^V \in \mathbb{R}^{D \times d_v}$. It is common to choose $D = d_q = d_k = d_v$, which means that $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are matrices in $\mathbb{R}^{N \times D}$. The attention mechanism can then be expressed as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax} \left(\frac{\mathbf{Q} \mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}. \quad (3.25)$$

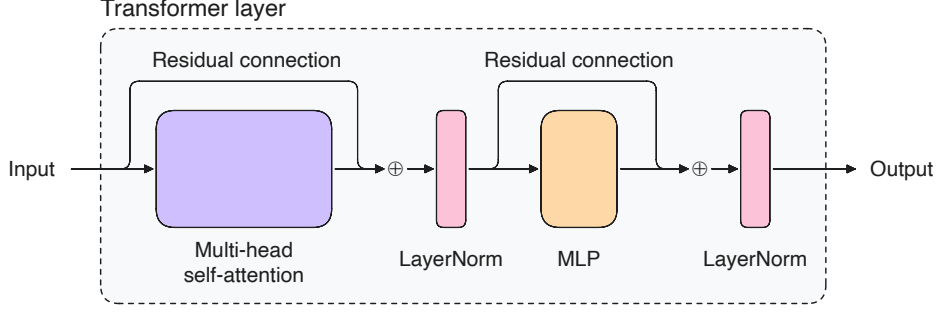


Figure 3.3: Forward pass through a transformer layer. The transformer layer combines the multi-head attention block with residual connections, layer normalization (LayerNorm), and an MLP that is applied point-wise.

The output of the self-attention calculation is a new matrix $\mathbf{X}' \in \mathbb{R}^{N \times D}$. Thus, self-attention transforms an input point cloud $\{\mathbf{x}_i\}$ into a new point cloud $\{\mathbf{x}'_i\}$ of the same cardinality N . Each vector $\mathbf{x}'_i$ in the new point cloud can also be expressed as a weighted sum of the value-vectors $\mathbf{v}_m = \mathbf{x}_m \mathbf{W}^V$ via

$$\mathbf{x}'_i = \sum_{m=1}^N A(\mathbf{x}_i, \mathbf{x}_m) \mathbf{v}_m, \quad (3.26)$$

where $A(\mathbf{x}_i, \mathbf{x}_m)$ denotes the *attention weight* between the i -th and m -th input vector. The attention weight $A(\mathbf{x}_i, \mathbf{x}_m)$ is calculated from the dot product between the corresponding query vector $\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^Q$ and key vector $\mathbf{k}_m = \mathbf{x}_m \mathbf{W}^K$, followed by the scaling by $1/\sqrt{d_k}$ and the Softmax function along the index m . The Softmax function ensures that the attention weights of a position i with respect to another position m add up to one when summing over all possible positions m .

It is common to apply multiple attention modules in parallel, which is called *multi-head attention*. Each head has its own parameters, and performs its computations independently of the other heads. Afterwards, the outputs for the h different heads are concatenated and projected with another matrix $\mathbf{W}^O$ to combine the output of the individual attention heads.

A typical *transformer layer* (or *transformer block*) combines multi-head self-attention with additional components as shown in Figure 3.3. First, the input is processed with a multi-head self-attention block. The output of this self-attention block is added to the original input (element-wise), which is known as a *residual connection* [167]. Next, this intermediate representation is processed with layer normalization [168], denoted as LayerNorm. This is followed by a point-wise MLP, i.e. the same MLP is applied to each of the N input vectors of the point cloud separately, also surrounded by a residual connection. Lastly, another layer normalization is applied, which returns the final output of the transformer block. Multiple transformer blocks are stacked in order to form the whole transformer. As of today, many variants of the transformer architecture exist. Some of those are just small adjustments to the design explained here, like for example Pre-LayerNorm Transformers [169], which place the layer normalization right before the multi-head attention and MLP components. Others involve the addition of new components, for example the addition of so-called *registers* introduced in the context of vision transformers [170], which involves appending additional points to the point cloud during the forward pass.

The concept of attention can also be used with two different input sets $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\{\mathbf{s}_1, \dots, \mathbf{s}_M\}$, denoted as matrices $\mathbf{X} \in \mathbb{R}^{N \times D}$ and $\mathbf{S} \in \mathbb{R}^{M \times D}$, respectively. This is known as *cross-attention*, and in this case the queries are calculated from $\mathbf{S}$, while the keys and

values are calculated from $\mathbf{X}$. Therefore, the output of cross-attention is a set of cardinality M in this case. The original transformer [164] combined self-attention and cross-attention in the context of language translation. Cross-attention is also used in transformer-based classifiers, originally introduced in the context of image classification [171]. This is also known as *class-attention*, and the main idea is to perform cross-attention between an input set and a learnable *classification token* that extracts information from the input set via the attention mechanism.

3.4 Generative Modeling

Generative models are models that learn to generate data samples that follow the same distribution as the samples they are shown during training. There are plenty of applications where this can be helpful, both for consumer products and in science. Generative models can be built in different ways. This section provides a brief introduction to the two generative modeling frameworks used in the research presented in this thesis: *flow matching* and *next token prediction*.

3.4.1 Flow Matching

Flow matching [172–174] is a method to train a *continuous normalizing flow* (CNF) [175] to map from an initial distribution p_{init} to a data distribution p_{data} . The brief introduction to flow matching here is based on the overview provided in Ref. [176]. Both p_{init} and p_{data} are distributions over $\mathbb{R}^D$, and the goal is to find a time-dependent velocity field that enables the transformation of the initial distribution into the data distribution. The time t is a continuous parameter $t \in [0, 1]$. At $t = 0$, the random variable $\mathbf{x} \in \mathbb{R}^D$ follows the initial distribution $\mathbf{x}_0 \sim p_{\text{init}}$ and at $t = 1$ it follows the data distribution $\mathbf{x}_1 \sim p_{\text{data}}$. Between $t = 0$ and $t = 1$, the random variable $\mathbf{x}_t$ follows a *probability path* p_t . The velocity field is modeled by a neural network $\mathbf{v}_\theta : \mathbb{R}^D \times [0, 1] \rightarrow \mathbb{R}^D$. This network is trained to approximate the velocity field $\mathbf{u}_t(\mathbf{x})$, which transforms $\mathbf{x}_t$ according to the ordinary differential equation (ODE)

$$d\mathbf{x}_t = \mathbf{u}_t(\mathbf{x}_t) dt, \quad (3.27)$$

where the subscripts in $\mathbf{u}_t$ represents the time-dependency of the velocity field. Once a good approximation $\mathbf{v}_\theta(\mathbf{x}, t)$ is found for the target velocity field $\mathbf{u}_t(\mathbf{x})$, this approximation can be used to generate samples by first sampling $\mathbf{x}_0 \sim p_{\text{init}}$ from the initial distribution and then integrating the neural ODE

$$\mathbf{x}_1 = \mathbf{x}_0 + \int_0^1 \mathbf{v}_\theta(\mathbf{x}_t, t) dt. \quad (3.28)$$

The initial distribution is typically chosen to be a normal distribution, though in general any distribution from which samples can be drawn can be used for p_{init} . To train $\mathbf{v}_\theta$, one has to first define the probability path p_t , which describes how the distribution transforms from $p_0 = p_{\text{init}}$ to $p_1 = p_{\text{data}}$. A *conditional probability path* is conditioned on a sample at $t = 1$ as $p_t(\mathbf{x} \mid \mathbf{x}_1)$ and fulfills the conditions

$$p_1(\mathbf{x} \mid \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1), \quad p_0(\mathbf{x} \mid \mathbf{x}_1) = p_{\text{init}}(\mathbf{x}) \quad (3.29)$$

such that for a given data point $\mathbf{x}_1$, the corresponding probability path morphs the initial distribution p_{init} into a δ -distribution at exactly that data point. The *marginal probability*

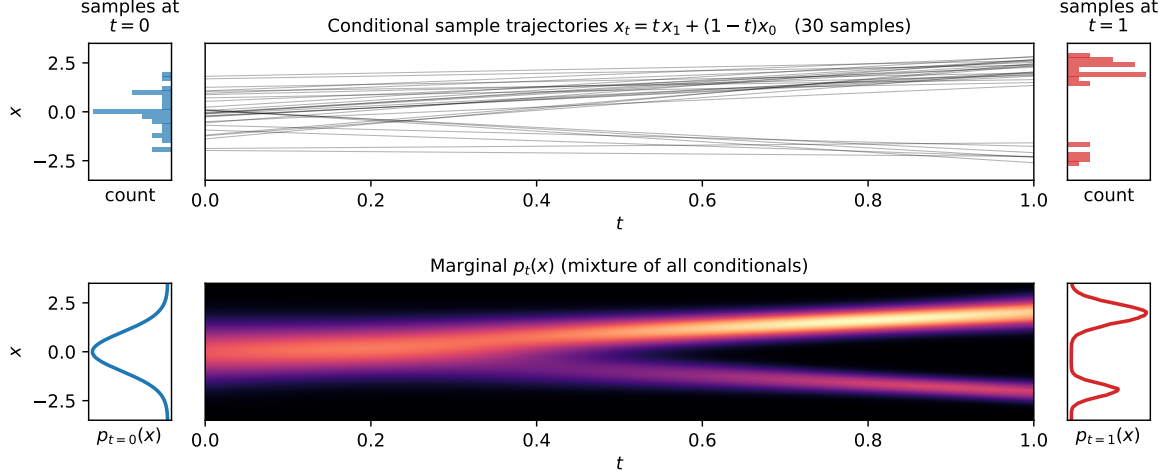


Figure 3.4: Illustration of the optimal-transport (OT) probability path for a 1D toy model. The initial distribution p_{init} ($t = 0$, left) is a standard Gaussian, and the data distribution p_{data} ($t = 1$, right) is composed of two separate peaks. The upper panel shows the conditional sample trajectories for 30 individual samples, as well as the corresponding histograms of x_0 and x_1 as projections on the left and right sides. The gray lines in the upper center panel represent the time evolution x_t of the individual samples. The lower panel illustrates the corresponding marginal probability path obtained by aggregating over many conditional probability paths, with the densities $p_{t=0}$ and $p_{t=1}$ shown as projections on the left and right side.

path $p_t(\mathbf{x})$ and the conditional probability path are related as

$$p_t(\mathbf{x}) = \int p_t(\mathbf{x} | \mathbf{x}_1) p_{\text{data}}(\mathbf{x}_1) d\mathbf{x}_1. \quad (3.30)$$

There are multiple variants of how the conditional probability path is constructed. A common choice is the *Gaussian conditional probability path*

$$p_t(\mathbf{x} | \mathbf{x}_1) = \mathcal{N}(\mathbf{x} | \alpha_t \mathbf{x}_1, \beta_t^2 \mathbf{I}), \quad (3.31)$$

where α_t and β_t are *noise schedules* with $\alpha_0 = \beta_1 = 0$ and $\alpha_1 = \beta_0 = 1$. With those conditions, the conditional probability path in Equation 3.31 satisfies Equation 3.29 with $p_{\text{init}} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ being a multivariate normal distribution. A popular choice for the noise schedules is given by the *conditional OT* or *linear* path:

$$\alpha_t = t, \quad \beta_t = 1 - t \quad (3.32)$$

$$\Rightarrow \mathbf{x}_t = t \mathbf{x}_1 + (1 - t) \mathbf{x}_0, \quad (3.33)$$

where $\mathbf{x}_t$ follows the distribution from Equation 3.31. This is visualized in Figure 3.4, which shows a one-dimensional example. The upper panel shows the straight trajectories $\mathbf{x}_t$ of individual samples given by Equation 3.33, whereas the lower panel shows the aggregated, marginal probability path.

As introduced earlier, the goal of flow matching is to find an approximation $\mathbf{v}_\theta$ of the velocity field $\mathbf{u}_t$ in Equation 3.27, such that $\mathbf{v}_\theta$ can then be used to draw new samples via Equation 3.28. The corresponding *flow matching loss* is given by

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t, \mathbf{x}_t} \|\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{u}_t(\mathbf{x}_t)\|^2 \quad (3.34)$$

$$\text{with } t \sim U[0, 1], \quad \mathbf{x}_t \sim p_t(\cdot),$$

where $U[0, 1]$ is the uniform distribution between zero and one. However, even with a simple choice for the probability paths, there is generally no closed-form expression for the corresponding $\mathbf{u}_t$, and thus the target in the above regression loss is unknown. A solution to this problem is given by *conditional flow matching (CFM)*, which formulates the task on a per-sample level.

Using the conditional OT probability path (Equation 3.32), one obtains a *conditional velocity field* satisfying the conditional analogue of Equation 3.27:

$$\mathbf{u}_t(\mathbf{x} \mid \mathbf{x}_1) = \frac{\mathbf{x}_1 - \mathbf{x}}{1 - t}. \quad (3.35)$$

With the conditional velocity field, the velocity field $\mathbf{v}_\theta$ can be trained via the *conditional flow matching loss*

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t, \mathbf{x}_t, \mathbf{x}_1} \|\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1)\|^2 \quad (3.36)$$

$$\text{with } t \sim U[0, 1], \quad \mathbf{x}_1 \sim p_{\text{data}}, \quad \mathbf{x}_t \sim p_t(\cdot \mid \mathbf{x}_1).$$

It can be shown that the *conditional flow matching* objective $\mathcal{L}_{\text{CFM}}$ (Equation 3.36) and the flow matching objective $\mathcal{L}_{\text{FM}}$ (Equation 3.34) have identical gradients, and thus the CFM objective allows training $\mathbf{v}_\theta$ to approximate $\mathbf{u}_t$ without knowing what exactly the unconditional $\mathbf{u}_t$ looks like [172]. Lastly, using Equations 3.33 and 3.35 corresponding to the conditional OT path, with $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, this loss becomes

$$\mathcal{L}_{\text{CFM}}^{\text{Gauss, OT}} = \mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_1} \|\mathbf{v}_\theta(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2. \quad (3.37)$$

The training of the velocity field then becomes surprisingly simple: samples are drawn from the training dataset and paired with samples that follow $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The time t is sampled as $t \sim U[0, 1]$, and the points $\mathbf{x}_t$ can be calculated according to Equation 3.33. Based on that, the velocity field $\mathbf{v}_\theta$ can be trained with the regression loss in Equation 3.37, and can subsequently be used to generate new samples that follow p_{data} using Equation 3.28.

3.4.2 Next Token Prediction

Next token prediction (NTP) [177, 178] is best known for its application in modern natural language processing [179–181], being the underlying mechanism of how modern language models work. However, the NTP formalism itself is not restricted to natural language. In this thesis, the NTP framework is used for generative modeling of sequences $(\mathbf{x}_1, \dots, \mathbf{x}_N)$, with the individual $\mathbf{x}_i$ being continuous-valued vectors $\mathbf{x}_i \in \mathbb{R}^N$. In NTP, it is assumed that the data can be represented as a sequence of integer-valued discrete *token-IDs*, denoted as t_i . The token-IDs can only take a finite number of values. The corresponding set of integer values $\{1, \dots, C\}$ is referred to as the *codebook*, with C being the *codebook size*. In the context of this thesis, the process of *tokenization* describes how the continuous vectors $\mathbf{x}_i$ are mapped to the corresponding token-IDs t_i , and how those token-IDs can be mapped back to the original feature space. The process of tokenization is described later in section 3.6. Once such a tokenization procedure is in place, a generative model can be built by training a model that generates new sequences of token-IDs, which are subsequently mapped to the original feature space.

An NTP-based model is an autoregressive model that factorizes the joint probability $p(t_1, \dots, t_N)$ of a sequence $(t_1, \dots, t_N)$ of length N as

$$p(t_1, \dots, t_N) = p(t_1) \prod_{n=2}^N p(t_n \mid t_1, \dots, t_{n-1}). \quad (3.38)$$

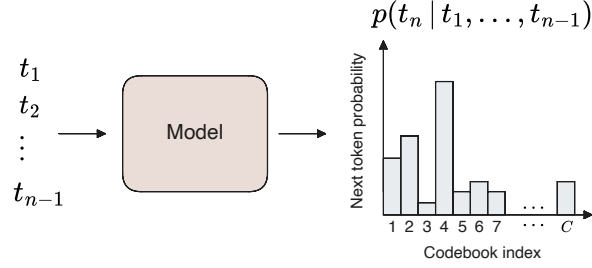


Figure 3.5: High-level illustration of NTP with codebook size C . For each of the C token-IDs in the codebook, the model predicts the probability of the next token corresponding to that codebook entry.

The task of the model is, as the name suggests, to predict each token t_n given the previous tokens $(t_1, \dots, t_{n-1})$. To this end, the model is trained to predict the conditional probability $p(t_n | t_1, \dots, t_{n-1})$. As the tokens can only take a finite number of values, this can be formulated as a multi-class classification problem with C classes, and can be trained with the cross entropy loss (see section 3.1.1). This is illustrated in Figure 3.5, where the model predicts the probability of the next token by returning the discrete probability distribution over the entire codebook.

Using this formalism, new sequences can be generated autoregressively. First, only t_1 is used as an input to the model, where in practice t_1 is a dedicated token that represents the beginning of a sequence, also referred to as the *start token*. The model then returns the probability distribution $p(t_2 | t_1)$ from which the next token t_2 can be sampled. This token is then appended to the sequence, which is now (t_1, t_2) . Next, this sequence is fed to the model to obtain $p(t_3 | t_1, t_2)$, from which in turn t_3 can be sampled and appended to the sequence. This procedure is repeated until a certain stopping criterion is met, often realized by defining a *stop token* that marks the end of a sequence or by setting a maximum sequence length at which the generation loop stops.

It is common to use a transformer architecture for an NTP model, with a minor modification to the attention mechanism introduced in section 3.3.2. This modified version of self-attention is called *causal self-attention*, and enables efficient¹ training of NTP models. The idea behind causal self-attention is to prevent a position i from attending to any “future” positions $j > i$. This is realized by adding a *causal mask* to the pre-Softmax attention weights, which sets the pre-Softmax attention weights for these positions to minus infinity. Consequently, the corresponding attention weights are effectively zero after applying the Softmax function. It should also be noted that causal self-attention is not permutation equivariant.

An example of a transformer for NTP is shown in Figure 3.6. The model can be split into three parts: (i) the input embedding layer, (ii) the transformer layers, and (iii) the linear output layer with a subsequent Softmax activation function. Neural networks are designed to operate on high-dimensional, continuous vectors, not on integer numbers. Therefore, the input embedding layer maps the token-IDs t_i into a high-dimensional representation in $\mathbb{R}^D$. This is achieved with a learnable lookup table, which stores such a high-dimensional representation for each codebook index. Next, the transformer layers process the sequence

¹Given the amount of computing resources needed to run modern LLMs, this might sound contradictory. Yet, the resource requirements originate from the corresponding model size, which was scaled to enormous extents, and with that the required compute to train and run those models. Causal self-attention itself is an efficient way of *training* NTP models as the whole sequence can be trained in parallel. During generation, however, this parallelism is lost, as the model then acts as an autoregressive model.

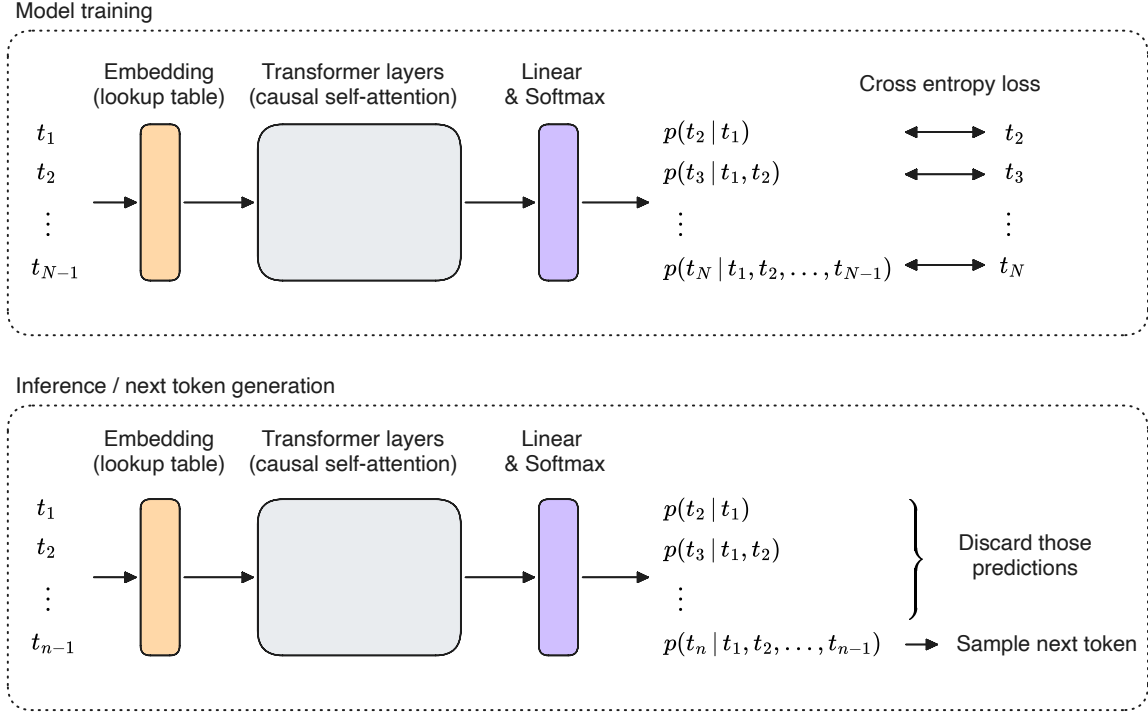


Figure 3.6: Illustration of the training (top) and inference (bottom) of an autoregressive generative transformer model based on NTP. During training (top), the whole sequence is processed in parallel, meaning that the next token predictions at all positions are included in the loss. During inference (bottom), only the next token prediction of the last position ($n - 1$ in this illustration) is used to predict the next token, and the predictions for the prior positions are discarded.

with causal attention. Finally, the output from the last transformer layer is passed through a linear layer followed by a Softmax activation function to obtain the next token probability distribution $p(t_n | t_1, \dots, t_{n-1})$ for each position n in the sequence. Those predictions are then compared to the one-hot encoded targets that represent the true next token.

During training (top panel in Figure 3.6), all predictions returned by the model are used for the loss calculation. That is, for an input sequence of length $N - 1$, the model returns $N - 1$ vectors with C entries. Each of these vectors represents the next token prediction probabilities for that position. The targets are obtained from the sequence itself, which makes this a self-supervised learning task: the input sequence $(t_1, \dots, t_{N-1})$ is mapped to the target $(t_2, \dots, t_N)$.

The model inference, i.e. using the trained model to predict next tokens one after the other in an autoregressive loop, is shown in the bottom panel of Figure 3.6. To predict the n -th token, the “partial” sequence $(t_1, \dots, t_{n-1})$ is used as input. The model produces a next token prediction for all $n - 1$ positions, ranging from $p(t_2 | t_1)$ up to $p(t_n | t_1, \dots, t_{n-1})$. This means that the model returns next token predictions for positions that are already generated. Thus, all predictions except for the last one, i.e. $p(t_n | t_1, \dots, t_{n-1})$, are discarded.

It should be noted, that it is not strictly required to use token-IDs as both the input and the target of the model. It is also possible to combine continuous-valued input with tokenized targets. Both cases, i.e. using token-IDs both as input and target, as well as a hybrid setup are studied in this thesis.

3.5 Masked Modeling

Masked modeling comprises methods where a model is trained to recover a masked portion of an input given the information that survived the masking process [182–186]. The main application of such methods is to learn an expressive representation of the input data. Representation learning can be very useful for pre-training an ML model before training it on the task it will then be used for. This will be discussed in more detail in chapter 4.

A way of implementing masked modeling is the *masked autoencoder* [186], which was originally introduced in computer vision, but can also be applied to other data representations like e.g. point cloud data [187]. A sketch of the model forward pass of a masked autoencoder is shown in Figure 3.7. Here, the input data is a point cloud $\{\mathbf{x}_i\}$ with $\mathbf{x}_i \in \mathbb{R}^D$ and D being the dimensionality of the vector of attributes associated with each point. Both the encoder and decoder shown in Figure 3.7 are assumed to be neural networks with a transformer architecture. Before being passed to the model, part of the input $\{\mathbf{x}_i\}$ is masked, e.g. 40 % in the example in Figure 3.7. Only the part of the input that survived this masking step is passed to the encoder of the masked autoencoder, returning $\{\mathbf{e}_j\}$ where j runs over the indices of the non-masked points. Next, the masked points are re-introduced to the point cloud in the form of *mask tokens* [182], or *mask embeddings*, $\{\mathbf{m}_k\}$ with k corresponding to the masked positions. The mask embeddings can be seen as the starting point from which the masked information is reconstructed, given the encoded, non-masked points as context. This new set, i.e. $\{\mathbf{e}_j\} \cup \{\mathbf{m}_k\}$, is passed to the *decoder* of the masked autoencoder, which tries to recover the information from the masked positions, with the recovered information being denoted as $\mathbf{x}'_k$. The mask embeddings $\mathbf{m}_k$ can be implemented as learnable vectors, and should contain partial information about the entity that is to be reconstructed at that position without trivializing the task itself. For example, in the original masked autoencoder, which presents masked modeling on images, each of the $\mathbf{x}_i$ represents an image patch [188], and both the encoder and decoder are composed of multiple transformer blocks. As the transformer architecture by itself is permutation equivariant, it has no inherent notion of the position of each element. For the model to be able to reconstruct masked image patches in a meaningful way, the mask token is implemented as a shared learned vector with additional positional encoding applied on top, i.e. an encoding that represents the position of the masked patch in the image.

The original masked autoencoder work [186] demonstrated this framework both for a

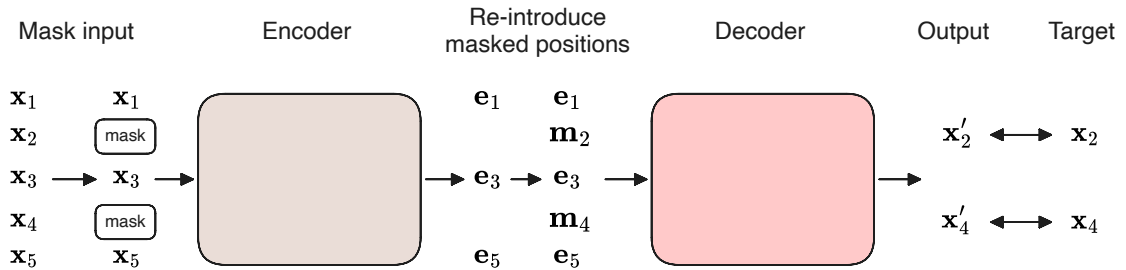


Figure 3.7: Sketch of a masked modeling setup in the style of a masked autoencoder. Part of the input $\{\mathbf{x}_i\}$ is masked as a first step. Next, the non-masked inputs $\{\mathbf{x}_j\}$ are processed with the encoder. The masked inputs are re-introduced after the encoder in the form of mask tokens $\mathbf{m}_k$. The new set $\{\mathbf{e}_j\} \cup \{\mathbf{m}_k\}$, i.e. the encoded, non-masked vectors and the mask tokens, is then processed with the decoder for subsequent reconstruction of the masked inputs. In this example, the reconstruction is phrased as a classification task with the targets being the token-IDs of the masked positions.

simple regression target, and for a tokenized target. Both options – regression on the masked vectors and prediction of the masked token-IDs – are used in the research presented in this thesis.

3.6 Tokenization with Vector-Quantized Variational Autoencoders

Tokenization is the process of finding a discrete set of tokens that can subsequently be used to represent the given data. Some data is by design discrete, such as natural language, and representing it via tokens is arguably the natural choice. In cases where the original data is continuous, learning a powerful mapping onto a discrete set of tokens allows using methods originally developed for discrete data, like NTP or masked modeling with tokenized targets. Here, the focus lies on the tokenization of continuous data relevant for the work presented in chapters 6 to 8. To this end, this section presents how tokenization of continuous data can be achieved with a vector-quantized variational autoencoder (VQ-VAE) [189]. The description here largely follows Ref. [190].

A VQ-VAE consists of an encoder F , a decoder G , and a codebook that stores the M -dimensional token embeddings $\{\mathbf{c}_1, \dots, \mathbf{c}_C\}$ of the token-IDs, with codebook size C . Loosely speaking, the overall aim of a VQ-VAE is to map a continuous input vector $\mathbf{x} \in \mathbb{R}^D$ to a discrete representation that can be reconstructed to an output vector $\mathbf{x}' \in \mathbb{R}^D$ with $\mathbf{x} \approx \mathbf{x}'$. The forward pass through a VQ-VAE is illustrated in Figure 3.8. First, the continuous input vector $\mathbf{x}$ is passed to the encoder, which returns the corresponding embedding $\mathbf{z}_e \in \mathbb{R}^M$ in an M -dimensional latent space. This latent space is the same space as the one where the token embeddings $\{\mathbf{c}_1, \dots, \mathbf{c}_C\}$ live. The token embeddings are illustrated in the sketch with dotted arrows. Within that space, the pairwise distances between the embedded feature vector $\mathbf{z}_e$ and the token embeddings $\mathbf{c}_i$ are calculated. Next, the feature vector $\mathbf{z}_e$ is mapped

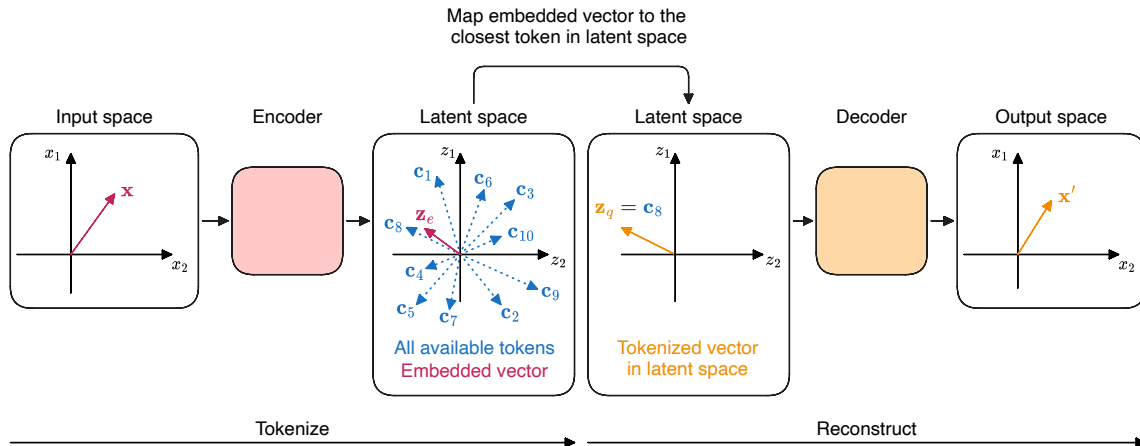


Figure 3.8: Schematic illustration of a forward pass through a VQ-VAE. The encoder transforms the input vector $\mathbf{x}$ into its latent space representation $\mathbf{z}_e$. This latent vector is mapped to the closest token embedding $\mathbf{c}_i$, which results in the quantized latent representation $\mathbf{z}_q$. This representation is decoded by the decoder network, yielding the reconstruction $\mathbf{x}'$. In this example, the input vector is mapped to $\mathbf{c}_8$, which means that it is assigned to token-ID = 8 during tokenization.

to the closest token embedding based on a distance measure $d(\cdot, \cdot)$, which yields the quantized embedding $\mathbf{z}_q \in \mathbb{R}^M$

$$\mathbf{z}_q = \mathbf{c}_k \quad \text{with} \quad k = \underset{i}{\operatorname{argmin}} d(\mathbf{z}_e, \mathbf{c}_i). \quad (3.39)$$

This quantized representation $\mathbf{z}_q$, which now corresponds to the token-ID k of the closest token embedding, is *decoded* by the decoder of the VQ-VAE. The decoder returns $\mathbf{x}'$ in the same space as the original input $\mathbf{x}$.

While the overall forward pass of a VQ-VAE is rather straightforward, it does require some tweaks for it to be trainable. The distance-based mapping in the latent space can be written as a quantization function $h : \mathbb{R}^M \times (\mathbb{R}^M)^C \rightarrow \mathbb{R}^M$, which takes $\mathbf{z}_e$ and $\{\mathbf{c}_1, \dots, \mathbf{c}_C\}$ as input, and returns $\mathbf{z}_q$ according to Equation 3.39. The distance measure is a hyperparameter of this mapping layer and can be e.g. the Euclidean distance. The whole forward pass can thus be written as

$$\mathbf{x}' = G(h(F(\mathbf{x}), \{\mathbf{c}_i\})) . \quad (3.40)$$

The VQ-VAE is trained to minimize the task loss $\mathcal{L}_{\text{task}}(\mathbf{x}', \mathbf{x})$, which quantifies the reconstruction quality of $\mathbf{x}'$. This task loss can be, for example, the MSE or MAE loss introduced in section 3.1.1. As the calculation of $\mathbf{x}'$ involves the non-differentiable quantization function h , this task loss is – without further modifications – not differentiable and thus the model cannot be trained with gradient descent. This can be bypassed with the *straight-through estimator* [191], which uses the pre-quantization embedding $\mathbf{z}_e$ for the gradient computation. In practice, this is realized by using $\mathbf{z}_q = \mathbf{z}_e + \text{stop_gradient}[\mathbf{z}_q - \mathbf{z}_e]$ in the forward pass, with `stop_gradient` being an operator that removes the corresponding gradients from the computation graph. However, this approximation is only valid if $\mathbf{z}_q$ and $\mathbf{z}_e$ are close together. Thus, another component is added to the overall loss of the VQ-VAE, with the aim of pulling the encoder representation $\mathbf{z}_e$ and the token embeddings $\mathbf{c}_i$ closer together. This loss component is called *commitment loss*, and is given by

$$\mathcal{L}_{\text{cmt}}(\mathbf{z}_e, \mathbf{z}_q) = (1 - \beta) \cdot d(\mathbf{z}_e, \text{stop_gradient}[\mathbf{z}_q]) + \beta \cdot d(\text{stop_gradient}[\mathbf{z}_e], \mathbf{z}_q). \quad (3.41)$$

The hyperparameter $\beta \in [0, 1]$ defines how much emphasis is put on adjusting the token embeddings towards the encoder representation or vice versa. The VQ-VAE can then be trained with the total loss

$$\mathcal{L} = \mathcal{L}_{\text{task}}(\mathbf{x}', \mathbf{x}) + \alpha \cdot \mathcal{L}_{\text{cmt}}(\mathbf{z}_e, \mathbf{z}_q), \quad (3.42)$$

where α is a scalar hyperparameter.

Tokenization of continuous data with a VQ-VAE inherently comes with some loss of information due to the finite number of codebook entries. Thus, it is important to choose appropriate network architectures for the encoder and decoder networks to keep this loss of information at an acceptable level for the application at hand. Additionally, while the discussion up to here only considered tokenization of a single continuous vector $\mathbf{x} \in \mathbb{R}^D$, this concept can also be applied to other types of data. The original VQ-VAE [189] work, for example, demonstrated this concept for images, videos, and audio [189]. Lastly, it should be noted that training a VQ-VAE can be challenging, especially due to so-called *codebook collapse* [190]. Codebook collapse means that only a small fraction of the codebook entries (i.e. the token embeddings) are actively used by the VQ-VAE. Ideally, all codebook entries are used in order to maximize the expressiveness of the model.

In the context of this thesis, VQ-VAEs are used to tokenize continuous data in order to be used for next- or masked token prediction. When used for an NTP-based generative

model, the model first predicts a sequence of token-IDs, which can then be decoded – also referred to as *reconstructed* here – to obtain the generated sequence in the original space of the data. No reconstruction is needed in masked modeling, as in this case the token-IDs only serve as targets for a proxy task for representation learning.

4 Foundation Models for Particle Physics

In recent years, foundation models (FMs) [26] have emerged as a powerful paradigm for building performant ML models for a variety of tasks. The term *foundation model* is best known from large language models, yet the concept goes well beyond natural language processing. This chapter introduces the concept of foundation models and motivates the overarching research question of this thesis: *how can one build a foundation model for particle physics?* First, an introduction to the overall concept of a foundation model is presented in section 4.1. This is followed by an overview of what a foundation model could look like in particle physics in section 4.2, and why particle jets are a suitable starting point for the development of HEP foundation models. Section 4.3 describes the public benchmark datasets used in the upcoming chapters. Finally, the different pre-training methods proposed for jet foundation models are reviewed in section 4.4.

4.1 Foundation Models

The term *foundation model* was coined in 2021 in Ref. [26], where a foundation model was defined as follows:

A foundation model is any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks; current examples include BERT [Devlin et al. 2019], GPT-3 [Brown et al. 2020], and CLIP [Radford et al. 2021]. [26, p. 3]

The concrete examples given in the above quote are large language models (BERT [182] and GPT-3 [181]), and a model that connects images and text (CLIP [192]). The foundation model paradigm is commonly visualized as in Figure 4.1, splitting the model training into two stages: *pre-training* and *fine-tuning*. The idea is to first train the model on a pre-training task and subsequently use the corresponding pre-trained model weights as the starting point for the training on a certain downstream task, with the latter being referred to as fine-tuning. This concept of *transfer learning* [193], i.e. that the capabilities a model has learned for a different task can be beneficial for a certain target task, was used long before the first foundation models appeared [194]. However, while transfer learning was studied before the rise of foundation models, it was found to be much more powerful once the corresponding models and the training dataset were *scaled* in size. As for the model size, this essentially

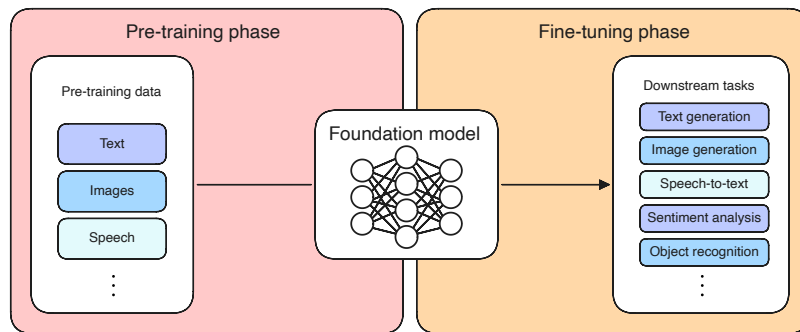


Figure 4.1: The foundation model paradigm as used for applications like NLP, computer vision, and speech processing.

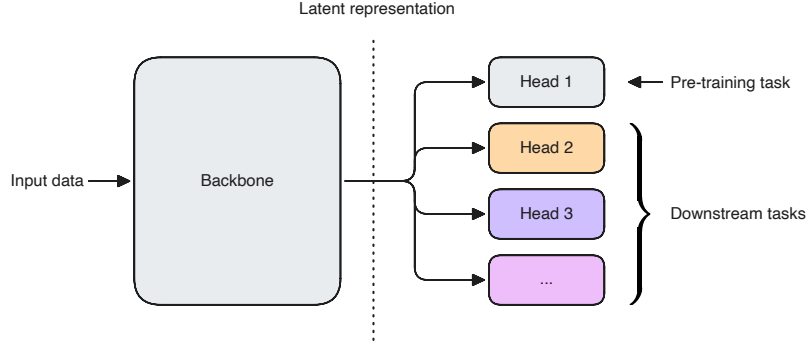


Figure 4.2: Illustration of the high-level view of a foundation model architecture used in this work. The core of the foundation model is called the *backbone*, which transforms the input into a latent representation that can be used for different tasks. Each task comes with a dedicated model *head*, which transforms the latent representation into the required output structure for the given task.

means that the number of parameters of a neural network is increased by using many, wide layers. In the case of the dataset size this simply means to use a larger dataset for model training. Furthermore, studies of foundation models found empirical *neural scaling laws* [195–197], which provide a quantitative framework that predicts how a model’s loss scales with the model size, the dataset size, and the amount of compute used for training.

In particle physics, the definition of a foundation model is commonly based on a generous interpretation of the original definition from Ref. [26], which is also used in this thesis. Thus, a foundation model is considered to be a model that fulfills the following criteria [198]:

- The model is pre-trained on a large dataset. The goal of this pre-training is to obtain a rich latent representation that can be used for other tasks.
- The pre-trained model can be adapted, i.e. fine-tuned, to a downstream task that is different from the pre-training task. This can be either a completely different task, or the same task on a different dataset.
- The fine-tuned model performs better than the same model architecture trained from a random initialization, also referred to as the *from scratch* baseline.

The corresponding neural network architecture can then be split into a *backbone* and separate task-specific model *heads* as shown in Figure 4.2. The backbone of the model transforms the input into a latent representation that is used as input for the different model heads. One of the tasks is chosen to be the *pre-training task*, which means that the model is pre-trained by optimizing the backbone and the corresponding model head for this specific task. After pre-training, the latent representation from the backbone can be used as input for the heads of the *downstream tasks*. During downstream training, the weights of the backbone are expected to be in a state that only needs minor adjustments for the new task, hence the name fine-tuning.

4.2 Jet Physics as a Testbed for Foundation Models in HEP

The LHC data analysis workflow, described in section 2.3.2, involves many steps from the initial measurement to a physics result. The HEP community has put extraordinary effort into replacing and/or complementing individual algorithms with a corresponding ML-based

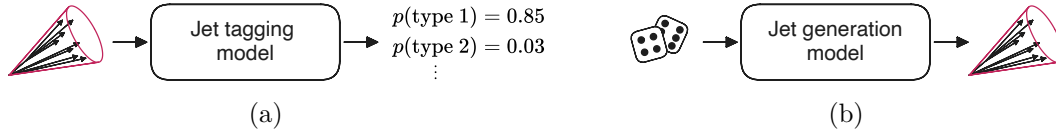


Figure 4.3: The fundamental idea of (a) jet tagging and (b) jet generation, respectively. A jet tagging model uses available information of the jet (usually constituent-level information like the constituent momenta) and returns a discrete probability distribution over jet types. A jet generation model generates new, realistic examples of particle jets based on a learned probability distribution.

counterpart, when applicable (see the review of ML developments for particle physics applications in Ref. [199]). However, this also means that a lot of resources – both in terms of time that goes into the development of individual ML algorithms and the corresponding compute power – are spent on ML development in HEP. The large number of potential downstream tasks highlights the suitability of the foundation model paradigm in particle physics. In addition to that, particle physics experiments record enormous amounts of data, which means that large datasets for foundation model pre-training are available. The ATLAS collaboration, for example, recorded around 18 billion complete events during Run 2 of the LHC, and generally produces 2–3 times more simulated events [200]. However, while the total amount of data is huge, specific physics analyses often require restrictive event selections, which can drastically reduce the dataset size available for the specific analyses. This also makes the foundation model paradigm an attractive direction for HEP: pre-train a foundation model on a huge, diverse dataset, and subsequently fine-tune the model for a specific analysis on a smaller dataset.

While the overall motivation for developing foundation models in HEP is apparent, it is not obvious which *kind of data* should be used as input for such a foundation model, and what an appropriate pre-training task would look like. For the latter, this might heavily depend on the choice of the data. As for the data, one could use the raw detector information (i.e. detector hits) or use reconstructed objects like particle jets. Moreover, the fact that both measured data and simulated data are available at the LHC raises the question whether the pre-training should be done on measured data directly, on simulated data, or both. The choice of measured data vs. simulated data also has an impact on the nature of the pre-training task: measured data is unlabeled whereas simulated data comes with labels that could be used for supervised pre-training tasks.

Over many years, jet physics has served the HEP community as a valuable benchmark for ML applications in particle physics [108]. This, combined with the fact that particle jets are of substantial relevance in particle physics, makes jet physics a well-suited testbed for the development of foundation models in this field. Common tasks involving jets are, among others, jet tagging [201–205], jet energy calibration [206, 207], track and vertex reconstruction [208], as well as combinations of jet tagging and jet generation in the context of anomaly detection [209–212]. In this thesis, the focus is on two of those tasks: *jet tagging* and *jet generation*.

Jet tagging, illustrated in Figure 4.3a, concerns the task of identifying the particle that initiated the jet. Thus, a jet tagging model takes as input information about the jet, which can be jet-constituent features or jet-level features, and uses that information to make a prediction of what type of particle led to the formation of the given jet. How the different classes are defined depends on the specific tagger: in the case of *boosted jet tagging*, the jet types can represent e.g. top quarks, $H/W/Z$ bosons, and background jets, also referred to

as QCD jets. The more fine-grained classification, which typically distinguishes between b -, c -, and light-flavor jets, is called *jet flavor tagging*. For the current state of the art of jet flavor tagging in the ATLAS and CMS experiments, see Refs. [201–205].

Jet generation, shown in Figure 4.3b, describes the task of generating realistic examples of particle jets. Here, a generative model learns to generate jets following the distribution from the dataset it was trained on. Such generative models can be used for applications in the search for new physics using anomaly detection methods [209–214].

4.3 R&D Benchmark Datasets

While experimental collaborations like ATLAS and CMS at the LHC have vast amounts of data (both measured and simulated) available, only a small fraction of that data is publicly accessible. For R&D work on ML methods for HEP, the community usually relies on public benchmark datasets. The benchmark datasets used in this thesis are described below. More details on the individual datasets can be found in the corresponding original publications.

4.3.1 Top Tagging Reference Dataset

The top tagging reference dataset [108, 109, 215] is aimed at the development of top tagging algorithms. It consists of two jet types: top jets, which are initiated by the hadronic top quark decay $t \rightarrow b q \bar{q}'$, and QCD jets, initiated by light quarks or gluons q/g . These jets are obtained from simulated LHC events at a center-of-mass energy of 14 TeV. The events are simulated with PYTHIA8.2 [91] followed by a simplified detector simulation with DELPHES [107] using the ATLAS detector card. Jets are clustered with FASTJET [128, 129] using the anti- k_t algorithm with $R = 0.8$. Only jets with $|\eta| < 2$ and $p_T \in [550, 650]$ GeV are stored in the dataset. Furthermore, top jets are required to have the parton-level top quark as well as the top decay products within $\Delta R < 0.8$ around the jet axis. For each jet, the kinematic information (i.e. the 4-momenta) of the jet constituents is stored in the dataset. This dataset contains 1.2 M jets for training, 400 k for validation and 400 k for testing.

4.3.2 The JetNet Dataset

The JetNet dataset [216, 217] was originally introduced in the context of generative modeling of jets, and has served as the standard benchmark dataset for this task for many years. JetNet is derived from the “HLS4ML LHC Jet dataset” [218], and contains simulated particle jets with $p_T \approx 1$ TeV extracted from proton-proton collisions at a center-of-mass energy of 13 TeV. It contains five jet types: gluon (g), light quark (q), top quark (t), W boson and Z boson. Of each jet type, the dataset stores around 175 k jets in total, with no default training/validation/test split. Parton-level simulation is performed with MADGRAPH5_AMC@NLO 2.3.1 [219], followed by parton showering with PYTHIA8.2 [91] and a simplified detector simulation. The jets are clustered with FASTJET [128, 129] using the anti- k_t algorithm with $R = 0.4$ ¹. As for the information stored in the dataset: it provides the jet p_T , jet η and jet mass, as well as the kinematic information of the constituents relative to the jet axis, i.e. $\Delta\phi$, $\Delta\eta$ and $p_{T,jet}$.

¹While the jet radius was reported as $R = 0.8$ in the original work [216, 217] where the dataset was introduced, Ref. [220] states that those jets correspond to $R = 0.4$. This stems from the fact that the η and ϕ distributions of the jet constituents do not align with the claimed $R = 0.8$, and the observation that the invariant mass distribution of the top jets shows a clear double-peak structure. The latter indicates that for a large number of top jets, only part of the top quark decay products are contained within the jet cone.

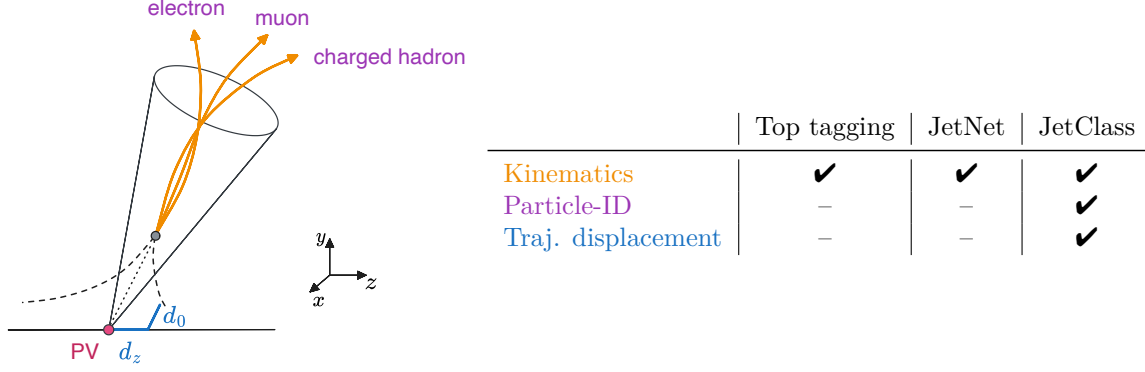


Figure 4.4: Illustration of the different feature categories stored in the JetClass dataset in comparison to the two other datasets discussed here. In addition to the kinematic information of the jet constituents, which is also given in the top tagging and JetNet datasets, this dataset provides an expanded feature set with kinematic, particle-ID, and trajectory displacement information. The depicted jet is for illustration purposes only, and not meant to represent a realistic example.

4.3.3 The JetClass Dataset

The JetClass dataset [110, 111] was introduced along with the Particle Transformer jet tagging algorithm [110], and represents a large scale dataset for R&D work in jet physics. It contains jets initiated by quarks or gluons (q/g) as well as jets arising from the decay of top quarks, W bosons, Z bosons, or Higgs (H) bosons. Jets associated with the decay of top quarks or Higgs bosons are further split into individual jet types depending on the decay channel: $t \rightarrow bq\bar{q}'$, $t \rightarrow b\ell\nu$, $H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, $H \rightarrow gg$, $H \rightarrow 4q$, $H \rightarrow \ell\nu qq'$, resulting in ten jet types in total. The production and decay of the t , Z , W , and H particles are simulated with MADGRAPH5_AMC@NLO [219], followed by parton showering and hadronization with PYTHIA8.2 [91]. Detector simulation is performed with DELPHES [107] with the CMS detector configuration. Jets are required to satisfy $500 \text{ GeV} < p_T < 1000 \text{ GeV}$ and $|\eta| < 2$. FASTJET [128, 129] is used to cluster the particles into jets with the anti- k_t algorithm and radius $R = 0.8$. For signal jets, i.e. all jet types except q/g , jets are required to fully contain the decay products of the particle that initiated the jet.

The feature set stored in the JetClass dataset is significantly more extensive than that of the top tagging dataset and the JetNet dataset, which only store the kinematic information of the jet constituents. Beyond the kinematic information of the particles, the JetClass dataset also includes particle-ID (PID) and trajectory displacement information. The PID is simply a flag that indicates the type of particle, i.e. if a particle is a photon, an electron, a muon, a charged hadron, or a neutral hadron. The trajectory displacement is visualized in Figure 4.4. This feature, also referred to as *impact parameter*, corresponds to the closest distance between an (extended) particle track and the primary vertex (PV), and is measured both in the transverse (xy) and longitudinal (z) direction. The transverse impact parameter d_0 represents the closest approach of the extended track (visualized as a dashed line in Figure 4.4) in the transverse plane. The longitudinal impact parameter d_z corresponds to the distance along the z -axis between the PV and the point where d_0 is evaluated. Trajectory displacement information is especially informative for the identification of jets involving b - or c -hadrons [221].

Beyond the more comprehensive feature set, the JetClass dataset also stands out with its size. With 100 M jets for training, 5 M for validation, and 20 M for testing, this dataset

is much larger than previous public benchmark datasets such as the top tagging dataset (2M) and the JetNet dataset (880k). This makes JetClass a commonly used dataset for pre-training in studies of foundation models for particle physics, whereas the other two are more commonly used as datasets for downstream tasks.

4.4 Pre-Training Methods for Jet Foundation Models

Different pre-training paradigms were proposed and studied for jet foundation models in recent years, with many methods being developed in parallel to the research presented in this thesis. A brief overview of those different methods is provided in the following.

Masked Modeling Here, the model is trained to reconstruct a masked fraction of the input given the non-masked input. In the context of jet foundation models, this means that the model predicts a masked set of jet constituents, given the non-masked constituents of that jet. Masked Particle Modeling (MPM) [222, 223] is a pre-training method for jets that builds on methods like BERT [182] from NLP as well as BEiT [185] and Masked Autoencoders [186] from computer vision. Other approaches are based on the Joint Embedding Predictive Architecture (JEPA) [224, 225] paradigm where masked entities are predicted in the latent space. First JEPA-based methods for jets are J-JEPA [226] and HEP-JEPA [227].

Generative Pre-Training Building on the success of generative pre-training in NLP [179–181], the concept of NTP-based pre-training is also adapted in the context of jet physics [2, 4]. Jets are represented as sequences of particles, and the pre-training task is to predict the next particle in the sequence. Additionally, a set-to-set flow matching method was proposed in Ref. [223], yet the authors only demonstrated its utility as a pre-training task and did not show the generative performance of such a model. Lastly, the OMNILEARN [210, 211] and OMNILEARNED [214] frameworks are partially pre-trained on jet generation, combined with supervised jet classification. The works on NTP-based pre-training are discussed in more detail in chapters 6 and 8.

Contrastive Learning Methods like JetCLR [228] and RS3L [229] adapt the concept of SimCLR [230] to particle jets. Here, an encoder maps positive pairs of jet augmentations together and negative pairs, i.e. augmentations of different jets, apart in latent space. JetCLR [228] uses augmentations like rotations and translations in the $\eta - \phi$ plane, as well as theory-inspired augmentations via soft and collinear splittings of particles, and was further studied as a pre-training task in Ref. [231]. RS3L [229], on the other hand, creates augmentations of each jet by re-running parts of the simulation chain described in section 2.3.2.

Self-Distillation Methods based on self-distillation typically use a student-teacher approach where the teacher is an exponential moving average (EMA) version of the student network. Distillation with No Labels (DINO) [232–234] is a popular self-distillation method in computer vision. The idea behind DINO is to train the student network to match the output distribution of the teacher network, with the input of the two networks being different augmented views of the same object. Renormalization group Invariance with NO labels (RINO) [235] applies this concept to jets with physics-inspired augmentations for the different views of a jet. In RINO, different views correspond to different stages of the k_T clustering process to imitate the reverse process of parton showering.

Supervised Pre-Training Supervised pre-training approaches are also explored in the context of pre-training jet foundation models. Jet tagging algorithms like ParT [110], LGatr [236, 237], and Sophon [238] show significant performance increases on small target datasets when pre-trained on a large dataset. In addition to that, as mentioned in the paragraph about generative pre-training, the OMNILEARN [210, 211] and OMNILEARNED [214] approaches involve joint pre-training on jet tagging (supervised) and jet generation (unsupervised).

The approaches listed up to here are designed for input at the jet constituent level and are the approaches most closely related to the results presented in this thesis. Further developments in the context of building foundation models for particle physics include joint fine-tuning of individual networks within an analysis pipeline [239], fine-tuning of ML-based particle-flow to new detector geometries [240], as well as methods that operate on higher-level objects or event-level approaches [241, 242].

Part II

Research Contributions

Overview

Within the overall theme of developing foundation models for particle physics, this thesis presents multiple research contributions. A short overview of the individual contributions is provided below, followed by summaries of the corresponding publications in chapters 5 to 9. The full length articles can be found in the *Publications* chapter at the end of this thesis.

Contribution 1: *Flow Matching Beyond Kinematics: Generating Jets with Particle Identification and Trajectory Displacement Information* (chapter 5) This contribution presents the first generative model trained on the JetClass dataset, based on the flow matching paradigm. While this contribution focuses on the generative task alone, it lays the foundation for follow-up studies of generative modeling on the JetClass dataset.

Contribution 2: *OmniJet- α : The First Cross-Task Foundation Model for Particle Physics* (chapter 6) Following the successful development of a generative model for the JetClass dataset in contribution 1, this contribution presents the first cross-task foundation model for particle physics by demonstrating transfer learning from the jet generation task to the jet classification task. This is achieved by exploring the GPT-paradigm, i.e. pre-training a generative transformer model with NTP.

Contribution 3: *Aspen Open Jets: Unlocking LHC Data for Foundation Models in Particle Physics* (chapter 7) Given the huge amounts of data recorded by the experiments at the LHC, this contribution studies if this data can be used for foundation model pre-training directly. This study is done with open data previously made public by the CMS collaboration. As part of this contribution, the corresponding dataset is released in an accessible, ML-friendly format to encourage the community to use this dataset for further studies based on particle jets from real, recorded LHC events.

Contribution 4: *Enhancing Next Token Prediction Based Pre-Training for Jet Foundation Models* (chapter 8) Building on the methodology introduced in contribution 2, this work presents further developments in the context of NTP-based pre-training for jet foundation models. The OmniJet- α model architecture is revised, leading to significant performance improvements in the downstream classification task. In addition to that, this work presents a joint pre-training method that combines NTP based pre-training with MPM [222, 223].

Contribution 5: *Pre-Training for Simulation-Based Science: A Study on Jet Foundation Model Training Objectives* (chapter 9) This contribution studies whether a HEP foundation model should be pre-trained in an unsupervised way, e.g. with a generative or masked modeling task, or with a supervised task like jet tagging. This work is based on the framework used to train the OMNILEARNED [214] foundation model, and provides a comprehensive comparison of different pre-training targets in the context of jet foundation models.

5 Flow Matching Beyond Kinematics [1]

This chapter summarizes the publication “Flow Matching Beyond Kinematics: Generating Jets with Particle Identification and Trajectory Displacement Information” [1]. The research was performed in collaboration with Erik Buhmann, Cedric Ewen, Gregor Kasieczka, and David Shih. This project is based on the original EPiC-FM [220] model which was developed by Cedric Ewen and Erik Buhmann. I adjusted the EPiC-FM model for the JetClass dataset in collaboration with Cedric Ewen, carried out all studies presented in the publication, and led the manuscript writing. The content of this chapter, including text, figures, and tables, is similar or identical to the original publication in Ref. [1].

In recent years, the task of jet generation has served as a valuable benchmark for ML developments in the HEP community. For the majority of those studies, the JetNet [216, 217] dataset (see section 4.3.2) has been the underlying benchmark dataset. Many generative modeling approaches have been studied with the JetNet dataset, including generative adversarial networks (GANs) [163, 217, 243–245], normalizing flows [246], diffusion models [210, 211, 247–249], and flow matching models [214, 220]. However, with around 175 k jets per jet type, the JetNet dataset is very limited in size. By contrast, the JetClass dataset [110, 111] (see section 4.3.3) stores a total of 12.5 M jets per jet type, and a more extensive feature set, which opens up new possibilities for the development of generative models for jets.

The model developed in this work is based on the Equivariant Point Cloud Flow Matching (EPiC-FM) [220] model, which was originally developed on the JetNet dataset. The model presented here is the first generative model developed on the JetClass dataset and also represents the first model that explores ML-based generative modeling of particle-ID and trajectory displacement features. The model is split into two individual parts: a model that generates jet-level features and a model that generates the jet constituents conditioned on those jet-level features. The conditioning features are the number of constituents, the jet transverse momentum, the jet pseudorapidity, and the jet types. The jet-level model is implemented with a kernel density estimator (KDE), whereas the constituent-level generative model is a continuous normalizing flow trained via conditional flow matching (see section 3.4.1). The constituent-level model, which transforms the jet constituents, uses EPiC [163] layers (see section 3.3.2), making it a permutation-equivariant CNF. Compared to the original EPiC-FM model in Ref. [220], this version of EPiC-FM is composed of 20 instead of 6 EPiC layers with increased hidden dimensionality. Another architectural change compared to the initial EPiC-FM model is the jet-type conditioning, which allows training a single model for all jet types simultaneously, while still being able to generate jets of a dedicated type of interest.

The quality of the generated jets is evaluated both on particle-level and on jet-level. For the particle-level features, this evaluation is split into the three feature categories mentioned in section 4.3.3: kinematic features, particle-ID features, and trajectory displacement features. The particle-level modeling of selected features from the different feature categories is shown in Figure 5.1, with the corresponding Kullback-Leibler (KL) divergence of the different features shown in Table 5.1 to quantify the agreement of the distributions. As an example of the kinematic features, the relative transverse momentum $p_T^{\text{rel}} = p_T/p_T^{\text{jet}}$, i.e. the transverse momentum p_T of the particle relative to the jet transverse momentum p_T^{jet} , is shown in Figure 5.1a. For better readability, the corresponding distributions of this feature are shown for q/g and $t \rightarrow bq\bar{q}'$ jets only. These two jet types are chosen for this comparison, as they show very different signatures in the p_T^{rel} feature. The q/g jets have on average

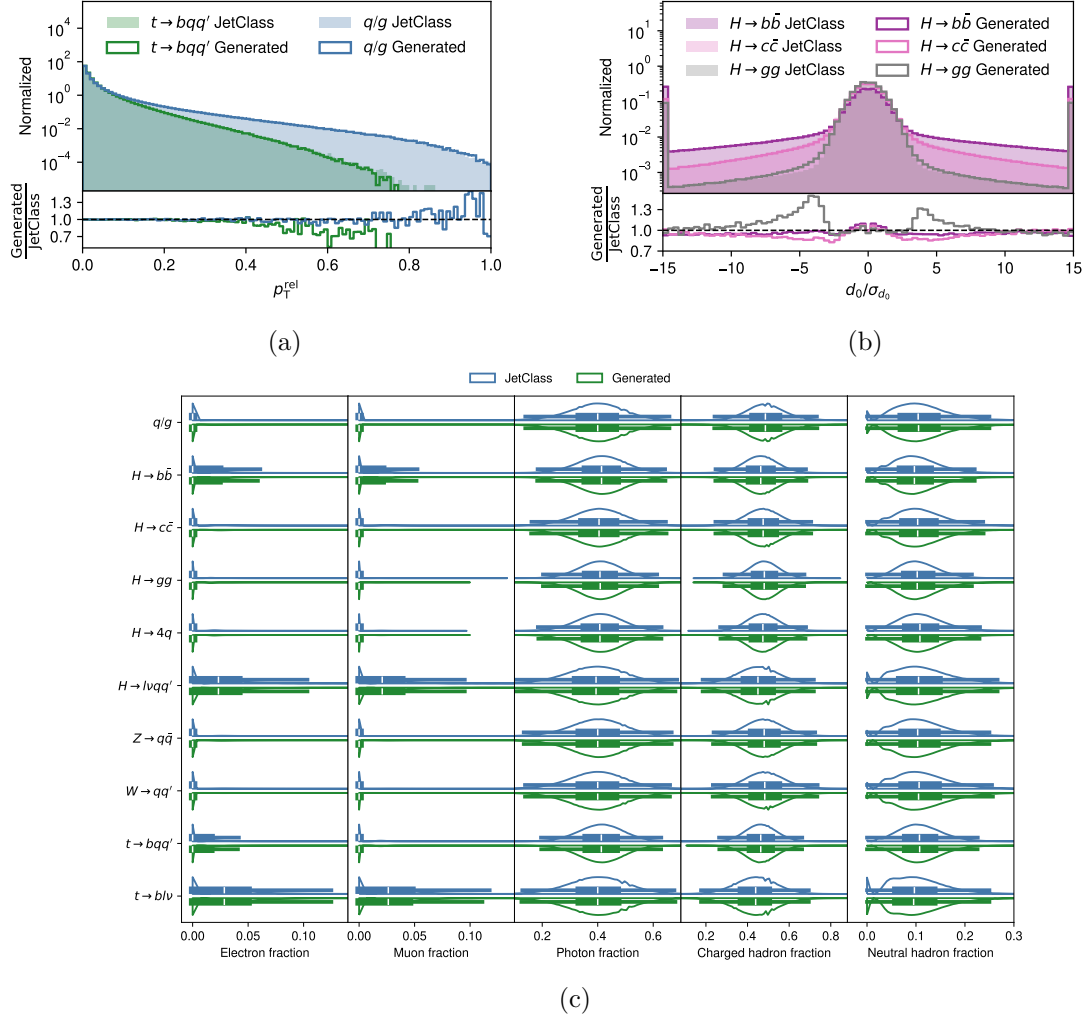


Figure 5.1: Particle-level modeling of (a) the relative transverse momentum p_T^{rel} (b) the transverse impact parameter significance of the associated particle tracks, and (c) the particle-ID features. The impact parameter is only shown for charged particles, as neutral particles do not leave a track in the tracking detector and thus their impact parameter is set to zero in the dataset. The histograms shown in (a) and (b) with solid lines correspond to jets generated by the EPiC-FM model. The semi-transparent histograms show the distributions obtained from jets from the JetClass dataset. The violin plots in (c) show the distribution of the fraction of particles within a jet corresponding to a certain type. The box plots in (c) show the median, the quartiles, and 1.5 times the interquartile range. These figures were previously published in Ref. [1].

fewer constituents than $t \rightarrow bq\bar{q}'$ jets, and with that have very different shapes in the relative transverse momentum distributions. The distributions obtained from the generated jets match the target distributions very well for both jet types, demonstrating that the model is capable of capturing those distinct signatures in the kinematic features. This can also be seen in the corresponding KL divergence values in Table 5.1, with only minor deviations from the reference. As a representative feature for the modeling of the trajectory displacement, the impact parameter significance, which is defined as the impact parameter value divided

Table 5.1: KL divergence of a selected set of features with respect to the JetClass dataset. Smaller values correspond to better agreement between the two distributions. The “Truth” reference compares two independent samples of jets from the JetClass dataset. The results in this table were previously published in Ref. [1].

	$\text{KL}^{p_{\text{T}}^{\text{rel}}} (\times 10^{-3})$	$\text{KL}^{d_0/\sigma_{d_0}} (\times 10^{-3})$	$\text{KL}^{m_{\text{jet}}} (\times 10^{-3})$	$\text{KL}^{\tau_{32}} (\times 10^{-3})$
Truth (q/g)	0.06 ± 0.01	0.13 ± 0.01	2.0 ± 0.2	2.0 ± 0.3
EPiC-FM (q/g)	0.073 ± 0.007	2.22 ± 0.06	2.0 ± 0.3	2.3 ± 0.3
Truth ($H \rightarrow b\bar{b}$)	0.048 ± 0.004	0.10 ± 0.01	1.9 ± 0.2	2.1 ± 0.2
EPiC-FM ($H \rightarrow b\bar{b}$)	0.070 ± 0.008	2.64 ± 0.09	3.2 ± 0.5	2.7 ± 0.3
Truth ($H \rightarrow c\bar{c}$)	0.054 ± 0.007	0.11 ± 0.02	2.0 ± 0.3	2.2 ± 0.3
EPiC-FM ($H \rightarrow c\bar{c}$)	0.078 ± 0.007	2.8 ± 0.1	4.8 ± 0.5	3.1 ± 0.5
Truth ($H \rightarrow gg$)	0.034 ± 0.002	0.07 ± 0.01	2.0 ± 0.2	1.9 ± 0.2
EPiC-FM ($H \rightarrow gg$)	0.057 ± 0.009	2.4 ± 0.1	3.7 ± 0.4	2.8 ± 0.4
Truth ($H \rightarrow 4q$)	0.040 ± 0.005	0.08 ± 0.01	2.2 ± 0.4	2.0 ± 0.3
EPiC-FM ($H \rightarrow 4q$)	0.057 ± 0.006	2.30 ± 0.07	5.2 ± 0.5	7.5 ± 0.7
Truth ($H \rightarrow \ell\nu qq'$)	0.07 ± 0.01	0.14 ± 0.01	1.9 ± 0.3	2.0 ± 0.2
EPiC-FM ($H \rightarrow \ell\nu qq'$)	0.10 ± 0.01	2.11 ± 0.08	3.5 ± 0.5	9.3 ± 0.9
Truth ($Z \rightarrow q\bar{q}$)	0.061 ± 0.008	0.14 ± 0.03	2.1 ± 0.2	2.0 ± 0.2
EPiC-FM ($Z \rightarrow q\bar{q}$)	0.07 ± 0.01	3.3 ± 0.2	4.8 ± 0.5	2.0 ± 0.2
Truth ($W \rightarrow qq'$)	0.06 ± 0.01	0.12 ± 0.01	2.1 ± 0.4	2.0 ± 0.2
EPiC-FM ($W \rightarrow qq'$)	0.082 ± 0.008	2.18 ± 0.05	5.7 ± 0.4	2.7 ± 0.4
Truth ($t \rightarrow bqq'$)	0.043 ± 0.007	0.08 ± 0.01	1.9 ± 0.3	2.1 ± 0.2
EPiC-FM ($t \rightarrow bqq'$)	0.08 ± 0.01	1.84 ± 0.07	4.7 ± 0.6	30 ± 1
Truth ($t \rightarrow b\ell\nu$)	0.07 ± 0.01	0.15 ± 0.02	2.0 ± 0.1	2.0 ± 0.2
EPiC-FM ($t \rightarrow b\ell\nu$)	0.11 ± 0.02	1.78 ± 0.09	3.4 ± 0.2	2.6 ± 0.4

by its uncertainty, is shown in Figure 5.1b. Both the value and the uncertainty are generated by the model, and are combined afterwards to obtain the significance. Here, three jet types corresponding to different decay channels of the Higgs boson are shown in the figure: $H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, and $H \rightarrow gg$. While those three jet types all represent jets initiated by a Higgs boson, the corresponding trajectory displacement distribution depends on the decay channel and thus varies between the three different jet types. $H \rightarrow b\bar{b}$ and $H \rightarrow c\bar{c}$ jets include hadrons with long lifetimes, which leads to secondary vertices with the associated tracks having large impact parameter values. Conversely, tracks in $H \rightarrow gg$ jets generally have smaller impact parameter values. These details and differences between different jet types are well modeled by the EPiC-FM model. However, as can be seen in the ratio panels of Figure 5.1b and in the KL divergence values in Table 5.1, the modeling of the trajectory displacement features is less accurate than that of the kinematic features.

The particle-ID modeling is shown in Figure 5.1c, which shows that the fraction of certain particle types within each jet is modeled accurately. This is non-trivial, especially given that those features are treated as continuous features throughout the generative modeling process, with a final post-processing operation that ensures that only the particle-ID feature with the largest value is set to 1, and the others to zero.

In addition to the constituent level evaluation, the fidelity of the generated jets is also analyzed on jet-level. To this end, jet-level quantities are calculated from the generated jet

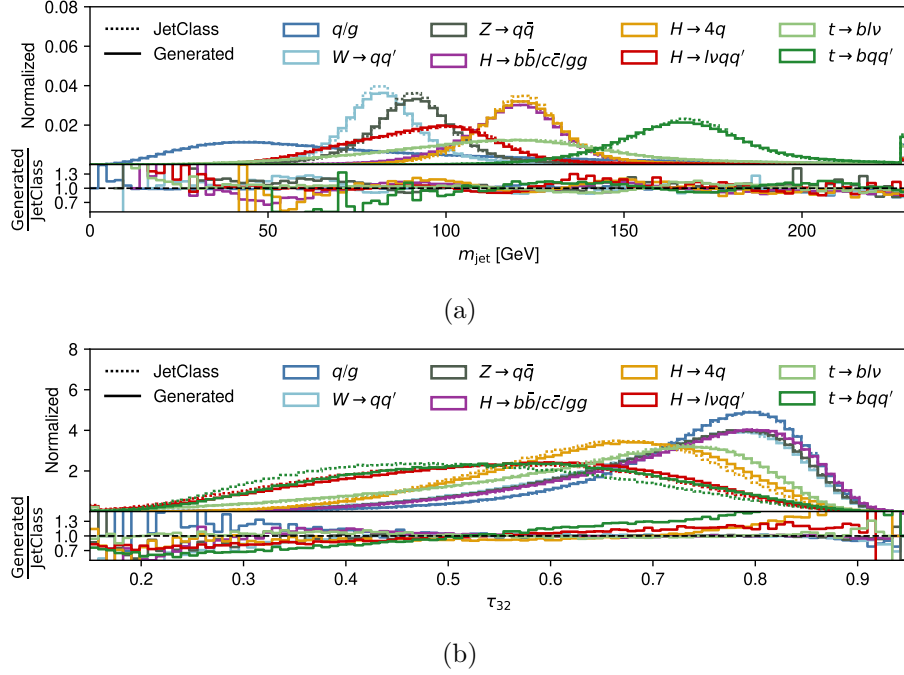


Figure 5.2: Comparison of two jet-level features for all jet types: (a) the invariant mass m_{jet} and (b) the subjettiness ratio τ_{32} . Each jet type is represented by a distinct color, with generated jets being shown as solid line histograms and jets from the JetClass dataset shown as dotted line histograms. The outermost bins show under- and overflow bins. The three jet types $H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, and $H \rightarrow gg$ are combined for better readability. These figures were previously published in Ref. [1].

constituents or the jet constituents from the JetClass dataset, respectively. Figure 5.2 shows the distributions of the jet mass and the subjettiness ratio τ_{32} (see section 2.3.3) for all ten different jet types, and the corresponding KL divergence values are shown in Table 5.1. This figure also shows how diverse the JetClass dataset is, with the jet mass spanning the whole range up to ~ 200 GeV, split into the individual mass peaks belonging to the corresponding heavy resonance ($Z/W/H/t$) of the non- q/g jets. Each of the jet mass peaks in Figure 5.2a is modeled with high accuracy. The most prominent mismodeling can be seen in the subjettiness ratio distributions in Figure 5.2b, where the largest disagreement between the generated distribution and the JetClass distribution is seen for $t \rightarrow bqq'$ jets. The observation that $t \rightarrow bqq'$ jets are the ones the model struggles the most with is further confirmed by a classifier test as proposed in Ref. [250], where a classifier is trained to distinguish jets from the JetClass dataset and jets obtained from the generative model.

To investigate this further, ablation studies are performed with a model trained on kinematic features only, a model that restricts the training dataset to $t \rightarrow bqq'$ jets only, as well as the combination of both restrictions. Those ablation studies reveal that a substantial improvement in the jet mass and jet substructure modeling is achieved when constraining the constituent-level feature set to kinematic features only. Both the jet mass and the jet substructure features are calculated from the kinematic features of the constituents. Therefore, this indicates that the modeling of the kinematic features improves drastically when the PID and trajectory displacement features are removed from the dataset, which in turn improves the quality of the jet mass and jet substructure modeling. Restricting the model to $t \rightarrow bqq'$

jets only is found to help when all features are used (i.e. kinematics, PID and trajectory displacement) whereas no improvement is seen with this restriction when the feature set is limited to kinematic features.

To summarize, this work demonstrates that the EPiC-FM model can be extended from the JetNet dataset to a more complex dataset like JetClass. Moving from the JetNet dataset to the JetClass dataset leads to: i) twice as many jet types, ii) significantly more constituent-level features, iii) larger-radius jets, and iv) more jets for training. Consequently, this work provides a foundation for further studies on generative modeling of jets from the JetClass dataset.

6 OmniJet- α [2]

This chapter summarizes the publication “OmniJet- α : The First Cross-Task Foundation Model for Particle Physics” [2]. The research was performed in collaboration with Anna Hallin and Gregor Kasieczka. The model development and the studies shown in this work were jointly performed by Anna Hallin and me. The writing process of the manuscript was distributed equally among the three co-authors of the paper. The content of this chapter, including text, figures, and tables, is similar or identical to the original publication in Ref. [2].

Inspired by the success of foundation models in fields like NLP or computer vision, this project explores the foundation model paradigm for particle jets. More precisely, this work focuses on generative pre-training with next token prediction (see section 3.4.2) as the pre-training target. While autoregressive generative modeling of particle jets was explored previously [251, 252], it was not used as a dedicated pre-training task with subsequent transfer learning to another task. This contribution represents the first successful demonstration of a prototype foundation model that shows transfer learning across different classes of tasks (i.e. classification and generation), with the corresponding model being named OmniJet- α .

The JetClass dataset (see section 4.3.3) is used throughout this project. For simplicity, the feature set is limited to the kinematic information of the particles. This builds on the result presented in chapter 5, which showed that, while in general possible, the modeling of trajectory displacement features can be challenging. Thus, in this study each particle is represented by the continuous vector $\vec{p} = (p_T, \eta^{\text{rel}}, \phi^{\text{rel}})$, with η^{rel} and ϕ^{rel} being the pseudo-rapidity and azimuth angle relative to the jet axis.

An overview of the overall framework used here is shown in Figure 6.1. This framework

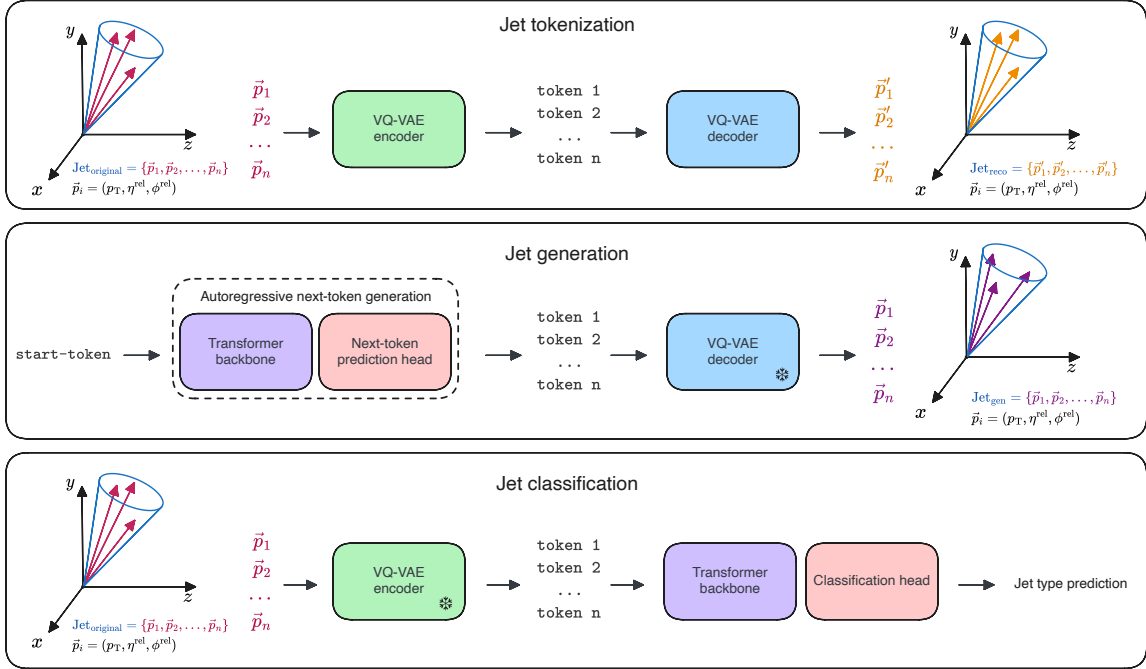


Figure 6.1: High-level overview of the OmniJet- α foundation model framework. This figure was previously published in Ref. [2].

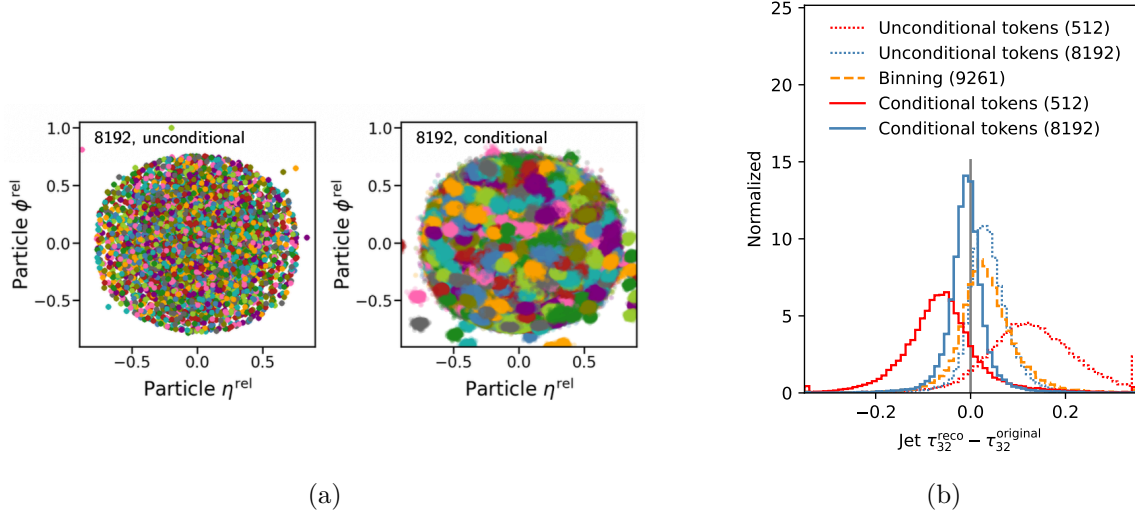


Figure 6.2: Token quality results: (a) visualization of how well the tokens cover the space and (b) the resolution of the subjettness ratio τ_{32} for $t \rightarrow bq\bar{q}'$ jets. The visualization in (a) shows the reconstructed codebook entries in the $\eta^{\text{rel}}\text{-}\phi^{\text{rel}}$ plane obtained with different codebook sizes and different tokenization strategies. The label in the upper left corner of each plot indicates the codebook size and the tokenization strategy. Each codebook entry, i.e. each possible value the token-IDs can take, is drawn as a colored dot in the scatter plot. To visualize the spread of individual codebook entries in the case of conditional tokenization, each codebook entry is reconstructed 500 times conditioned on 50 randomly chosen other codebook entries. The τ_{32} resolution in (b) shows the difference between the subjettness ratio τ_{32} of the original jet and the same quantity calculated from the reconstructed tokens. These figures were previously published in Ref. [2].

has three components: i) the tokenization of the jet constituents, ii) the generative model obtained from pre-training, and iii) downstream training on jet classification (i.e. jet tagging).

The transformer backbone used for the generative model and the classification model operate on discrete token-IDs (see section 3.4.2). Thus, the continuous-valued jet constituents are discretized with a VQ-VAE [189] (see section 3.6). The tokenizer is trained on all ten jet types of the JetClass dataset simultaneously. As mentioned in section 3.6, tokenization inherently comes with a loss of information. To investigate the extent of this information loss, this work compares three different tokenization approaches: a simple binning approach as used in Ref. [251], unconditional tokenization which uses a simple MLP for both the encoder and the decoder of the VQ-VAE, and the conditional tokenization strategy proposed in Ref. [222]. The latter uses a transformer architecture for the encoder and decoder of the VQ-VAE. With the binning approach and the unconditional tokenization, each token-ID corresponds to a distinct point in the phase space spanned by the kinematic features of the constituents. The reason for this is that each constituent is tokenized and reconstructed independently. In conditional tokenization, the tokenization and reconstruction of a constituent is conditioned on the other constituents in the jet [222]. As a result, each codebook entry can be reconstructed to multiple points in phase space. This can be seen in Figure 6.2a, which shows the spread of the token-IDs in the $\eta^{\text{rel}}\text{-}\phi^{\text{rel}}$ plane for the unconditional and conditional VQ-VAE tokenizer. Conditional tokenization provides the highest-fidelity encoding of the jet constituents. This is shown in Figure 6.2b, which shows that the subjettness ratio resolution improves drastically, when using the conditional tokenization approach. Increasing the codebook size also improves the subjettness resolution. The binning approach

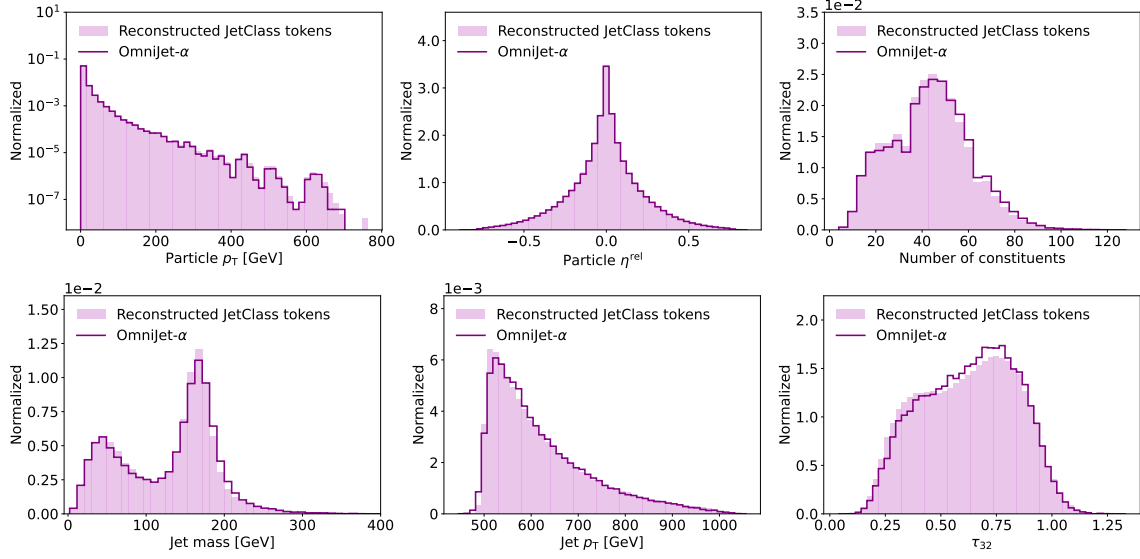


Figure 6.3: Generative modeling performance of the OmniJet- α model on particle level (top) and jet level (bottom). The reference histograms correspond to jets from the JetClass dataset that are first tokenized and subsequently reconstructed back to physical space, as this is the target the generative model is trained on. This figure was previously published in Ref. [2].

performs worse than the other two approaches. Using the conditional tokenization strategy with a codebook size $C = 8192$, the dataset is tokenized and a jet with n constituents is subsequently represented as a sequence of $n + 2$ token-IDs

$$(\text{start-token}, t_1, \dots, t_n, \text{stop-token}), \quad (6.1)$$

with dedicated token-IDs marking the beginning (**start-token**) and the end (**stop-token**) of the sequence.

The main model follows the high-level model architecture scheme introduced in section 4.1, being composed of a transformer backbone and task-specific heads, with one head being a generative head and the other a classifier head. The backbone consists of an embedding layer that maps the integer token-IDs into a high-dimensional embedding space, followed by three transformer blocks. The generative head is a linear layer followed by a Softmax activation function to obtain the next token probabilities. The classifier head consists of two linear layers, with a ReLU activation function and summation over the particle dimension between the two linear layers, and finally a Softmax activation function.

Using the tokenized particle representation from Equation 6.1, the transformer backbone is pre-trained as an autoregressive NTP model (see section 3.4.2). That is, the model learns to predict the token-ID of the next particle in the sequence. This pre-training is performed jointly on $t \rightarrow bq\bar{q}'$ and q/g jets (20M jets). Once trained, the generative model can generate new token-ID sequences autoregressively. The generated token-IDs are then reconstructed with the frozen VQ-VAE decoder to map the token-IDs to the physical space. Both particle level and jet level distributions of the generated jets are compared with the corresponding target distribution in Figure 6.3. The distributions of the generated jets align very well with the target distributions, with only slight mismodeling seen in the jet mass and the subjettness ratio τ_{32} .

To investigate the transfer learning capabilities of the backbone, the generative head is replaced by the classification head. The classification task is performed on the same jet types

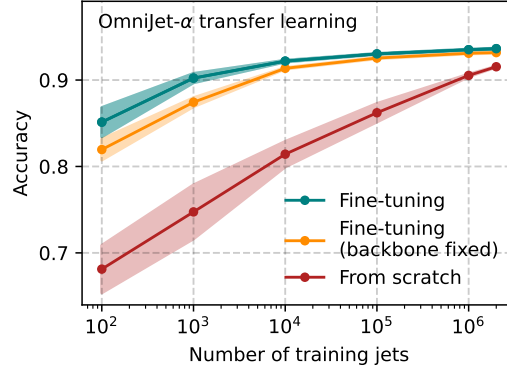


Figure 6.4: Transfer learning results evaluated on the top tagging task. For each configuration, the classification training is performed five times with different random seeds. The figure shows the mean performance and the corresponding standard deviation of those runs. The same pre-trained backbone is used in all fine-tuning runs. This figure was previously published in Ref. [2].

as used during pre-training and thus represents the typical top tagging task, i.e. the model learns to distinguish between $t \rightarrow bq\bar{q}'$ and q/g jets. Three different strategies are studied. In the *Fine-tuning* strategy the whole model is trained during downstream training. The *Fine-tuning (backbone fixed)* strategy keeps the backbone weights fixed during the classifier training, and the *From scratch* baseline represents a reference of the classification performance obtained with the same model architecture when all model weights are randomly initialized at the start of the training. For each strategy, classification training is performed for six different sizes of the downstream dataset, ranging from 100 jets to 2 M jets. The classification accuracy obtained for the different strategies and dataset sizes is shown in Figure 6.4. The two fine-tuning strategies lead to significantly better downstream performance than the from scratch baseline for the whole range of probed dataset sizes. Within the two fine-tuning strategies, the one that allows the backbone weights to be adjusted during downstream training performs the best for all dataset sizes. The performance gain from pre-training is most notable at small downstream dataset sizes: with only 100 training jets during classification training, the pre-trained model gains up to 17 percentage points in the accuracy over the from-scratch baseline. As the downstream dataset size increases, the performance gains of pre-training become smaller. Yet, the fine-tuned model still outperforming the baseline by a few percentage points at a downstream dataset size of 2 M. To summarize, this marks the first demonstration of cross-task transfer learning from unsupervised generative pre-training to supervised jet classification training. Along with other label-free pre-training methods (see section 4.4) this work shows promising potential for pre-training foundation models in HEP on measured data directly.

7 Aspen Open Jets [3]

This chapter summarizes the publication “Aspen Open Jets: Unlocking LHC Data for Foundation Models in Particle Physics” [3]. The research was performed in collaboration with Oz Amram, Luca Anzalone, Darius A. Faroughy, Anna Hallin, Gregor Kasieczka, Michael Krämer, Ian Pang, Humberto Reyes-Gonzalez, and David Shih. My contribution to this project was the training and evaluation of the tokenizers, the pre-training of the generative model, and writing the corresponding paragraphs in the manuscript. The content of this chapter, including text, figures, and tables, is similar or identical to the original publication in Ref. [3].

The original OmniJet- α publication discussed in chapter 6 demonstrated successful transfer learning between two different tasks on the same dataset (JetClass). This work follows up on those results, answering two main questions. First, it investigates if transfer learning can also be achieved for the jet generation task between two datasets with significant domain shift, i.e. two datasets with significantly different distributions. And second, it explores if such transfer learning can be realized when pre-training directly on measured data from an LHC experiment. Additionally, the corresponding ASPEN OPEN JETS (AOJ) dataset, which is derived from open data from the CMS collaboration [253, 254], is released in an accessible, ML-friendly format [255], representing the largest public jet dataset for ML development in particle physics.

Particle physics experiments like the CMS experiment at the LHC produce copious amounts of data. Performing foundation model pre-training on this measured data directly is attractive for two reasons. On the one hand this data already exists, and it could reduce the amount of computationally expensive simulated data (see section 2.3.2) needed for future foundation model training in HEP. On the other hand, while the simulation tools used in particle physics provide high fidelity simulation, there is still a remaining domain shift between simulated data and measured data. Most ML tasks in the HEP analysis pipeline require a labeled dataset in their training, and simulated data will remain crucial for high-performance ML models in particle physics. However, it could be advantageous to expose those models to measured data already during pre-training.

In general, measured data is only accessible to members of the corresponding collaboration of the experiment. Yet, a significant amount of measured data has been released as open data by the collaborations at the LHC [253, 254, 256–259]. Pre-training on open data was previously explored in Ref. [260], however not at the scale investigated in this work. The AOJ dataset is constructed from the CMS 2016 “JetHT”¹ open data [253, 254]. The open data is available in the MINIAOD [261] format that already includes CMS reconstruction steps (see section 2.3.2). Based on that, the AOJ dataset is produced by processing the initial MINIAOD files to the so-called PFNANO format via the corresponding CMS-provided tool [262], and subsequently extracting the jets from those events. The final dataset contains 178 M jets clustered with the anti- k_T algorithm with $R = 0.8$. For each jet, up to 150 constituents are stored, including features of multiple categories: kinematic features, particle-ID, trajectory displacement, as well as features corresponding to CMS-specific observables like flavor tagging scores. The vast majority of those jets are expected to be q/g jets. Still, the AOJ jets are very different from JetClass q/g jets in that they are selected with different kinematic

¹The JetHT dataset includes events that were selected based on one or more jets with high p_T , multiple low- p_T , or a significant amount of hadronic activity.

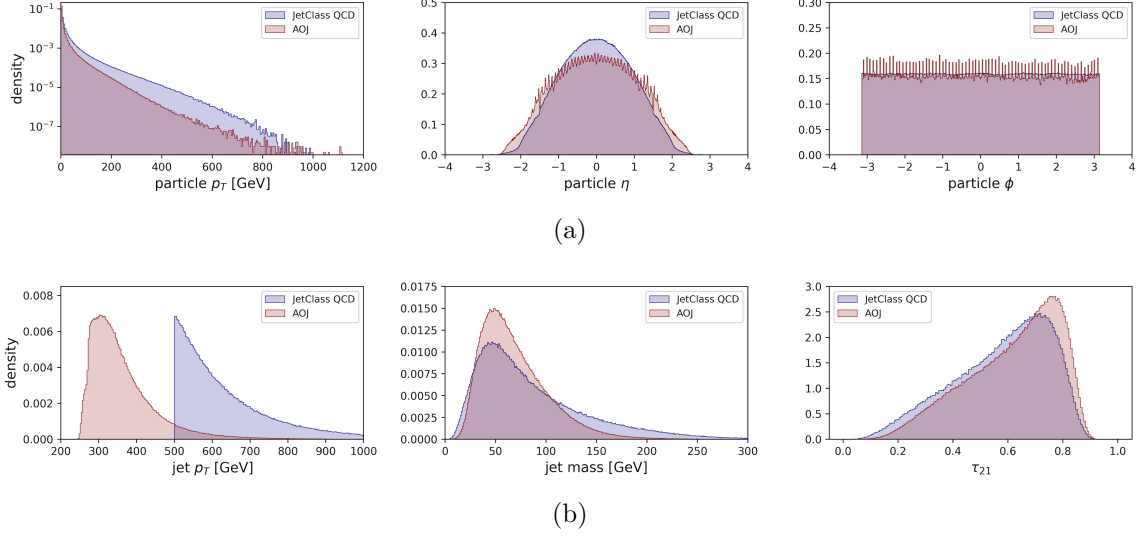


Figure 7.1: Feature distributions of jets from the AOJ dataset compared to q/g jets from the JetClass dataset (referred to in the label as “JetClass QCD”). These figures were previously published in Ref. [3].

cuts. A comparison of the AOJ jets and q/g jets from the JetClass dataset is shown in Figure 7.1 for a few selected features. The particle-level features (Figure 7.1a) demonstrate detector artifacts in the form of spikes in the pseudorapidity η and the azimuth angle ϕ of the particles. Such spikes do not show up in the JetClass dataset. As for the jet-level features (Figure 7.1b), those reveal that the distribution of the subjettness ratio τ_{21} , as well as the overall shape of the mass distribution of the AOJ jets is indeed consistent with q/g jets. At the same time the jet p_T distribution underlines the domain shift between the two datasets: there is only very little overlap between the two jet p_T distributions in Figure 7.1b, with jets from the JetClass dataset carrying a much higher transverse momentum.

To investigate the out-of-domain transfer learning, the OmniJet- α model is pre-trained for the generative task on the AOJ dataset, and this model is then fine-tuned to either q/g jets or $t \rightarrow bq\bar{q}'$ jets from the JetClass dataset. The transfer to $t \rightarrow bq\bar{q}'$ jets represents the more challenging case. This is because this transfer not only includes a difference in kinematic cuts between the AOJ and JetClass datasets, but also a drastic change in the jet substructure given that the AOJ dataset is assumed to contain mostly q/g jets. Two tokenizers are used in this study. For studies involving pre-training, a new tokenizer is trained on the AOJ dataset, which is then used to tokenize both the AOJ and the JetClass dataset. Both the pre-training and the fine-tuning are then performed using the token-IDs obtained with this tokenizer. For the from scratch baseline, which is trained directly on the JetClass dataset without pre-training, the tokenizer from the original OmniJet- α work is used. All downstream trainings on JetClass are repeated for different downstream training dataset sizes $D \in \{10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$. Figure 7.2 shows the downstream performance for $t \rightarrow bq\bar{q}'$ jet generation for two jet-level features: the jet mass and the subjettness ratio τ_{32} . Roughly speaking, the model has to learn the top quark mass peak and switch from generating 1-prong jets to 3-prong jets (see section 2.3.3 for an illustration) which shows up as a much smaller average τ_{32} value.

The plots on the left side of Figure 7.2 show the distributions obtained from the generative models, with different colors corresponding to the different downstream dataset sizes D . The plots on the right side show the corresponding Wasserstein distance W_1 and the KL di-

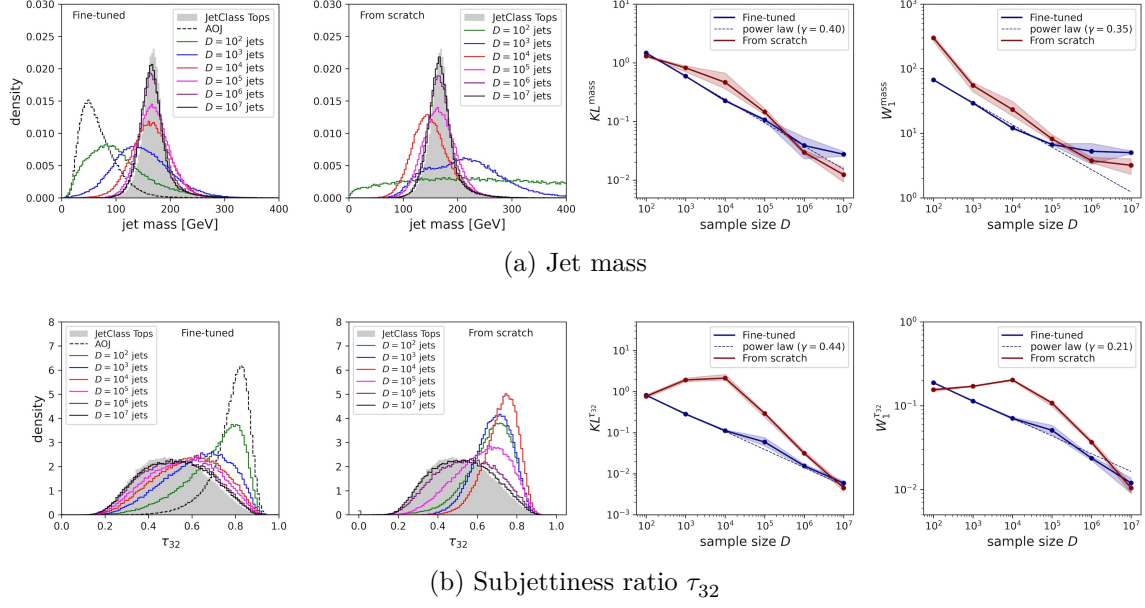


Figure 7.2: Comparison of the fine-tuned generative model versus the from scratch baseline for $t \rightarrow bq\bar{q}'$ jets. The top panel (a) shows the jet mass and the bottom panel (b) shows the subjeettness ratio τ_{32} . The leftmost plots show the distributions obtained when fine-tuning the model that was pre-trained on AOJ to the corresponding JetClass jet type. The second plot from the left shows the distributions produced by the from scratch baseline. The different histograms correspond to different downstream dataset sizes D , the AOJ distribution is shown as a dashed histogram, and the distribution of jets from the JetClass dataset is shown as a gray histogram. The two plots on the right side show the Wasserstein distance and the KL divergence calculated with the JetClass jets and the generated distribution for each downstream dataset size D . For the W_1 distance and the KL divergence values, the mean and standard deviation are reported by running each downstream training configuration with five different seeds. Smaller values of the W_1 and the KL divergence correspond to better agreement between the two distributions. These figures were previously published in Ref. [3].

vergence to quantify how well the distributions of the generated jets match the distributions from the JetClass dataset. Overall, the pre-trained model achieves a performance gain in that it needs only a fraction of the downstream training dataset size D to yield comparable performance to the baseline models. For large dataset sizes the from scratch baseline catches up with the fine-tuned model, indicating that the pre-training advantage vanishes at sufficiently large dataset scales. A possible explanation for this is that the tokenizer trained on the AOJ dataset is suboptimal for the JetClass dataset, and this starts to show up once such high-performance regimes are reached. Another possible explanation is that the pre-trained state is suboptimal if a sufficiently large downstream dataset is available, as was also seen e.g. in Ref. [196]. Looking at the jet mass modeling (Figure 7.2a), one can see that the pre-trained model manages to generate a clear peak correctly centered at the top quark mass of around 173 GeV with only 10 k training jets. The from-scratch model needs a factor of ten larger D to achieve that. Increasing the dataset size further improves the resolution of this mass peak, both for the pre-trained and the from-scratch model. As for the subjeettness ratio, the fine-tuning morphs the AOJ distribution into the top jet distribution, with better agreement the larger the dataset size D . This trend is not as consistent for the from-scratch model, and the pre-trained model shows better agreement with the JetClass distribution for

all D except for the full dataset size $D = 10$ M. An exception is the KL and W_1 at the smallest dataset size. At this dataset size, however, both pre-trained and from-scratch performs very poorly as can be seen in the histograms, and a lower KL or W_1 does not necessarily indicate better modelling in that case. Lastly, it is observed that the metrics for fine-tuned models follow a power law of the form $D^{-\gamma}$, with the corresponding numerical fit drawn in the plots in Figure 7.2. Such neural scaling laws [195–197] are closely related to foundation models, as they provide a framework for how a model’s performance scales with the dataset size, the number of trainable parameters, and the training cost, and thus such scaling laws can be used to study how a foundation model’s performance scales with those properties. Here, the pre-trained model shows a rather consistent scaling with $0.2 < \gamma < 0.45$ among different high-level features such as the jet mass and the subjettiness ratios. However, no such scaling is seen in this study for the models that are trained from scratch. This lack of scaling is most visible in the subjettiness ratio features in the low dataset size regime. The lack of scaling in those cases could be explained by the fact that the from-scratch model can hardly learn anything based on the small dataset it is provided during training. Such behavior was previously seen in the ML literature, and was termed the *small data region* [263]. Recent work dedicated to neural scaling laws of jet generation, which follows up on this, can be found in Ref. [264]. Further studies on neural scaling laws for ML models in HEP applications explore this framework for jet classification [203, 265–267], the pre-training loss of a foundation model [268], and amplitude surrogates [269].

To summarize, this work demonstrates that generative pre-training on measured collider data improves downstream jet generation performance on small labeled datasets, even with significant domain shifts between the pre-training and downstream datasets. Furthermore, the AOJ dataset is released in an ML-friendly format to encourage the community to explore the possibilities of using open data for phenomenological studies like this one.

8 Enhancing Generative Pre-Training for Jet Foundation Models [4]

This chapter summarizes the publication “Enhancing Next Token Prediction Based Pre-Training for Jet Foundation Models” [4]. The research was performed in collaboration with Anna Hallin, Gregor Kasieczka, Nikol Madzharova, Ian Pang, and David Shih. My contribution to this project was the development and implementation of the models, performing all studies presented in the paper, and large parts of the writing of the manuscript. The content of this chapter, including text, figures, and tables, is similar or identical to the original publication in Ref. [4].

The original OmniJet- α [2] work (see chapter 6) demonstrated that NTP-based generative pre-training leads to performance improvement in a downstream classification task compared to training a model with the same architecture from scratch. Additionally, follow-up work [3] (see chapter 7) demonstrated that the generatively pre-trained model can be fine-tuned efficiently to new datasets. This work improves on the original OmniJet- α implementation by addressing two of its limitations. First, it is shown that using the token-ID representation as the model input is suboptimal for downstream classification. To circumvent this limitation, this work introduces a hybrid setup that uses continuous input vectors while still using discrete token-IDs as pre-training targets. This leads to a drastic increase in classification performance while maintaining the generative capabilities of the model. Second, the pre-training task is enhanced by employing a joint pre-training that trains on NTP and masked particle modeling [222, 223] simultaneously.

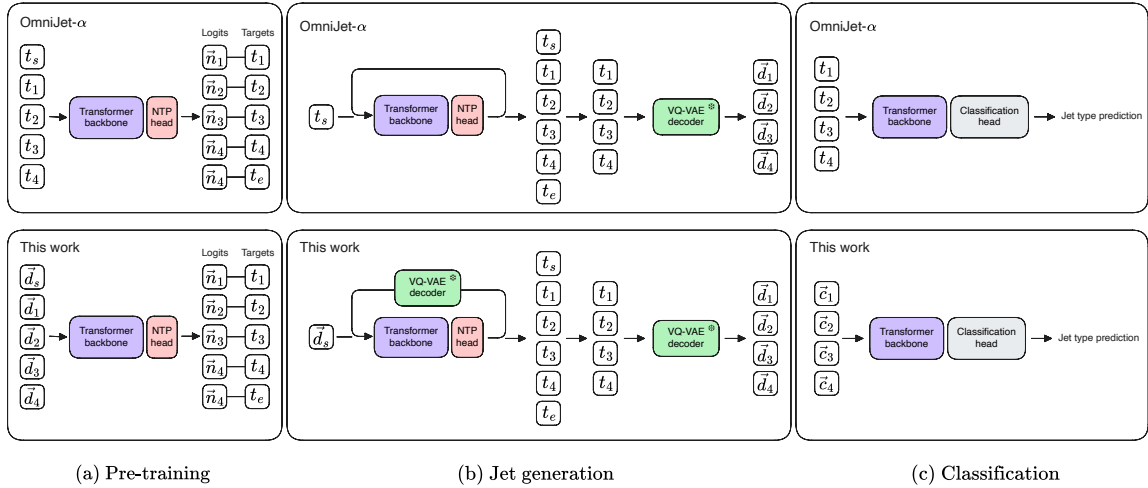


Figure 8.1: Comparison of the original OmniJet- α workflow (upper row) and the improved setup used in this work (lower row). The different columns show (a) the NTP-based pre-training, (b) the generative model obtained from unsupervised pre-training and (c) the classification model obtained from supervised fine-tuning. Token-IDs are indicated as t_i , with the start and end of the sequence being marked with start (t_s) and stop (t_e) tokens. Continuous feature vectors are shown as $\vec{c}_i$ and pseudo-continuous vectors as $\vec{d}_i$, where pseudo-continuous vectors are the ones obtained by tokenizing and reconstructing the original continuous input. This figure was previously published in Ref. [4].

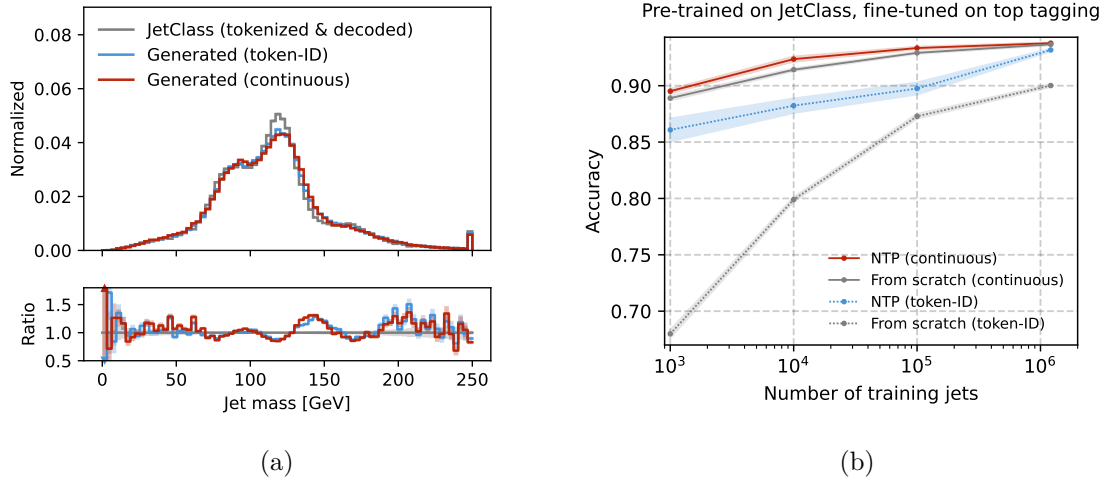


Figure 8.2: Comparison of using the token-ID or the continuous input vectors as model input: (a) the jet mass distribution obtained from the two generative models compared to the JetClass distribution and (b) the classification performance on the top tagging dataset. The uncertainty band in (b) corresponds to the mean performance and the corresponding standard deviation across five classification trainings with different random seeds. These figures were previously published in Ref. [4].

The general framework and architectural change in terms of the input structure compared to OmniJet- α (integer token-IDs vs. continuous) are shown in Figure 8.1. The change from using the integer token-IDs t_i as model input to using the corresponding continuous input vectors $\vec{c}_i$ as input has a clear advantage for the classification task, as this task is then not limited by any tokenization-induced information loss. During generative pre-training, the model acts on *pseudo-continuous* vectors $\vec{d}_i$ which correspond to the tokenized and reconstructed counterpart of $\vec{c}_i$. This distinction accounts for the fact that the decoding of the predicted token-IDs of the next particle during autoregressive generation (see the illustration in the center of Figure 8.1) is now integrated into the autoregressive generation loop. Using the full-resolution vectors $\vec{c}_i$ as input for the generative model would lead to a mismatch between training and inference (= generation) since only the pseudo-continuous vectors are available during generation.

To compare this new hybrid setup to the fully tokenized case, models of both architectures are first pre-trained on the generative task using the whole JetClass dataset (see section 4.3.3) and subsequently fine-tuned on the 10-class classification task on JetClass (in-distribution transfer learning) and on the binary classification task on the top tagging dataset (out-of-distribution transfer learning, see section 4.3.1). The architecture of the classifier head used here is based on class-attention [171], which is a more capable architecture compared to the simple classifier head architecture used in Ref. [2]. Given that the top tagging dataset only contains the kinematic information of the particles, the main studies of this work are performed with kinematic features only. First, the generative capabilities of the model are investigated both on particle level and on jet level. Figure 8.2a shows a comparison of the jet mass distribution obtained from the two generative models and the JetClass dataset, as this is a challenging distribution to model accurately. Note that, by contrast to the results presented in chapter 6, all ten jet types included in the JetClass dataset are used here, which is why the jet mass distribution here looks notably different from the one in Figure 6.3. Both models agree equally well with the target distribution, with slight mismodeling around the Higgs boson mass at approximately 125 GeV. This means that the switch to the hybrid

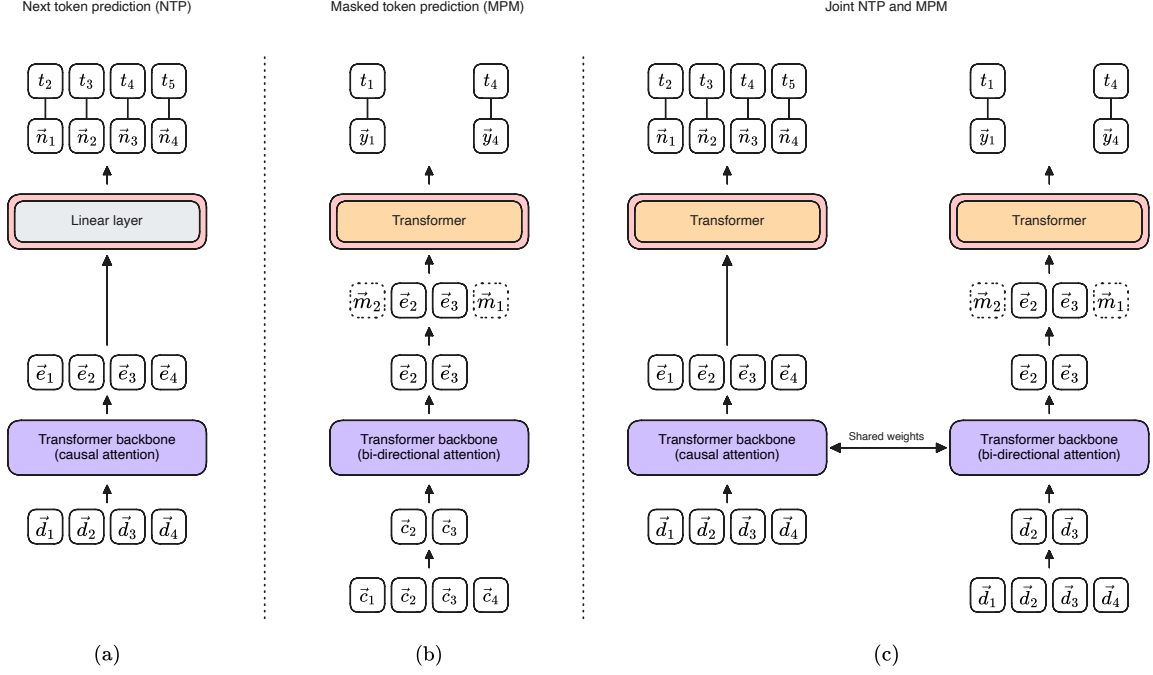


Figure 8.3: Illustration of (a) NTP, (b) MPM and (c) the joint NTP+MPM pre-training methods. Token-IDs are shown as t_i , continuous vectors as $\vec{c}_i$, and pseudo-continuous vectors as $\vec{d}_i$. The backbone output vectors are shown as $\vec{e}_i$ and the learnable mask embeddings that are re-introduced after the backbone in MPM are shown as $\vec{m}_i$. No sorting is applied to the particles, but masked positions contain information of the p_T -order within the masked subset. In this example, the first masked particle (position 1) is assumed to have smaller transverse momentum than the second masked particle (position 4), which is why they are re-introduced as $\vec{m}_2$ and $\vec{m}_1$, respectively. This figure was previously published in Ref. [4].

architecture does not harm the generative performance. The results for out-of-distribution transfer learning are shown in Figure 8.2b. Both pre-trained models are fine-tuned for multiple downstream dataset sizes $D \in \{10^3, 10^4, 10^5, 1.2 \cdot 10^6\}$. The corresponding from-scratch baselines for both input variants are also shown in Figure 8.2b. Those baselines (the two gray lines in Figure 8.2b) already show the expected increase in performance (here the accuracy) when switching from the token-ID input to the continuous input. This suggests that the classification model struggles to extract the required information from the token-IDs, leading to worse performance compared to the continuous input case. In terms of transfer learning capabilities, the gap between the performance of the fine-tuned model and the from-scratch baseline becomes smaller in the continuous-input case. However, generative pre-training on the NTP target still leads to improved downstream classification performance compared to training a model from a random weight initialization.

With the adjusted setup, the input-target structure is close to the MPM setup, which uses continuous input vectors and optionally tokenized targets. In addition to that, MPM and NTP are quite closely related as they predict either next or masked particles given other particles as context. The methodology of the joint NTP+MPM pre-training proposed in this work is shown together with the individual methods in Figure 8.3, with the MPM implementation being based on MPMv2 [223]. An important difference between NTP and MPM is that NTP requires causal attention, thus treating the input as a sequence, whereas MPM can use bidirectional self-attention treating the input as a point cloud. Combining these two

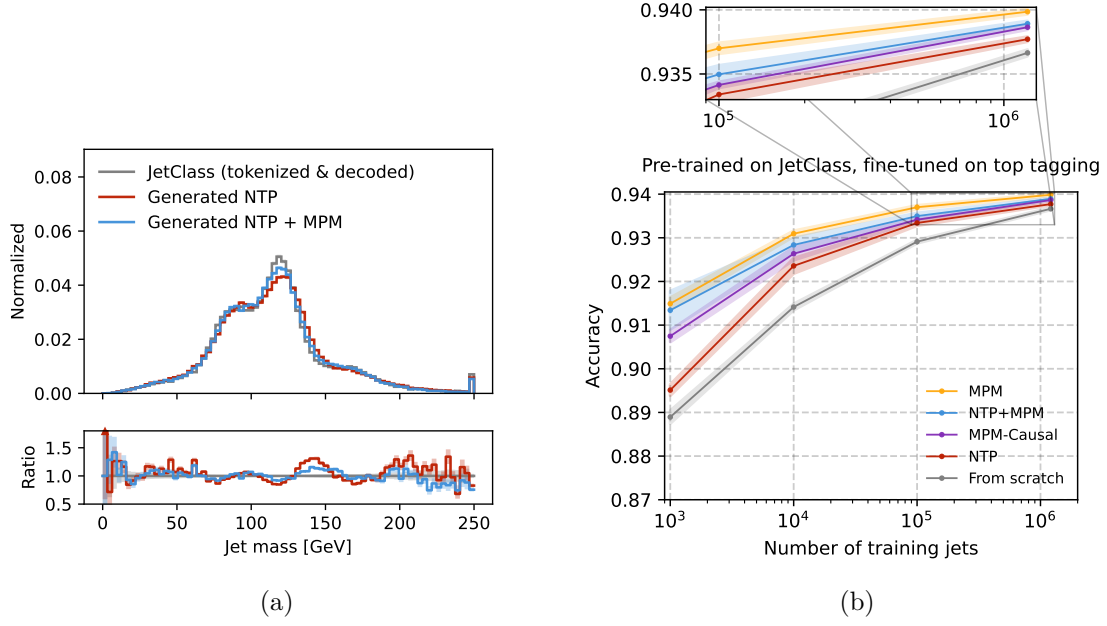


Figure 8.4: Comparison of the different pre-training methods: (a) the jet mass distribution obtained from continuous-input NTP and NTP+MPM and (b) the downstream top tagging accuracy achieved when pre-training with MPM, NTP or NTP+MPM. The MPM-Causal line represents an ablation where causal attention is used in the MPM pre-training. The uncertainty band in (b) corresponds to the mean performance and the corresponding standard deviation across five classification trainings with different random seeds. These figures were previously published in Ref. [4].

methods was previously studied in other domains [270–276]. As for the joint pre-training presented here, the hypothesis is that causal attention provides powerful embeddings for autoregressive generation that might not be perfectly aligned for the downstream classification task, whereas the bidirectional encoding learned during MPM pre-training might be better aligned with downstream classification. Combining the two methods could yield a rich latent representation with improved downstream performance on classification while at the same time maintaining the generative capabilities of the model. Figure 8.4a compares the jet mass distribution of the continuous-input NTP model and the joint NTP+MPM with the corresponding distribution from JetClass. This distribution shows that the generative model’s performance is retained in the joint pre-training. For this specific feature, the agreement with the distribution from JetClass even slightly improves compared to the NTP-only case. Other features are investigated as well and confirm that the generative performance does not degrade when adding the MPM component to the pre-training. The comparison of the downstream top tagging performance is shown in Figure 8.4b. The continuous-input NTP performance as well as the baseline are the same as in Figure 8.2b. The new lines in Figure 8.4b show the joint NTP+MPM as well as pure MPM and an ablation labeled MPM-Causal, with the latter demonstrating the effect of switching from bidirectional self-attention to causal self-attention. While NTP leads to a performance improvement compared to the from-scratch baseline, the other methods outperform NTP. The best downstream classification performance is achieved with MPM pre-training. The fact that the result obtained with MPM-Causal pre-training is worse than the non-causal MPM case indicates that using causal attention during pre-training harms the downstream classifier performance (which operates with bidirectional attention). Additional studies are performed with a fixed backbone dur-

ing fine-tuning, which indicate that the backbone representation obtained from NTP-only pre-training is substantially less aligned with classification compared to the representation obtained with MPM or NTP+MPM pre-training. The joint NTP+MPM pre-training clearly outperforms pure NTP pre-training, and even comes close to the top tagging performance of the model that was pre-trained with pure MPM. Given this clear advance in downstream classification, combined with the maintained generative capabilities of the model, the joint pre-training thus provides a substantial improvement over the NTP pre-training.

To sum up, this contribution presents two enhancements for NTP-based pre-training of jet foundation models that largely improve the downstream classification performance while maintaining the generative performance of the pre-trained model.

9 A Comparison of Jet Foundation Model Pre-Training Objectives [5]

This chapter summarizes the publication “Pre-Training for Simulation-Based Science: A Study on Jet Foundation Model Training Objectives” [5]. At the time of writing this thesis, the paper was published as a pre-print on arXiv, and is currently under review at PRX Intelligence. The research was performed in collaboration with Ibrahim Elsharkawy, Gregor Kasieczka, Vinicius Mikuni, Wahid Bhimji, and Benjamin Nachman. My contribution to this project was the integration of the MPM method into the OMNILEARNED framework, performing all MPM-related pre-trainings as well as the studies related to the generative downstream task. Additionally, I was responsible for the writing of the corresponding sections in the manuscript of the publication. The content of this chapter, including text, figures, and tables, is similar or identical to the original publication in Ref. [5].

The methods presented in chapters 6 to 8 focused on unsupervised pre-training. Unsupervised pre-training is the default choice in industrial foundation models due to the lack of large labeled datasets. While unsupervised pre-training methods are attractive for HEP as well in the sense of pre-training on measured data directly, particle physics has large datasets of high-fidelity simulation available. This led to supervised pre-training strategies like those used for the OMNILEARN [210, 211] and OMNILEARNED [214] foundation models. Those foundation models are pre-trained jointly on jet generation and jet classification, yet the original works did not investigate how the two targets compare individually. This work uses the OMNILEARNED framework to systematically compare three different pre-training targets: jet classification, jet generation, and MPM.

The jet generation task is implemented with conditional flow matching (see section 3.4.1) and the MPM implementation is based on MPMv2 [223]. In addition to the evaluation of the individual pre-training tasks and their effects on downstream performance, this work also studies the pairwise combinations (Classifier+Generator, Classifier+MPM, Generator+MPM) and the combination of all three targets. For some combinations, a perturbed loss component is added, corresponding to classification performed with the post-masking input (when combining classification and MPM) and classification with noised input, i.e. the input the generator forward pass receives (when combining classification and generation). The multiple forward passes corresponding to the different tasks and their associated loss terms are shown in Figure 9.1. All models are pre-trained on the JetClass dataset (see

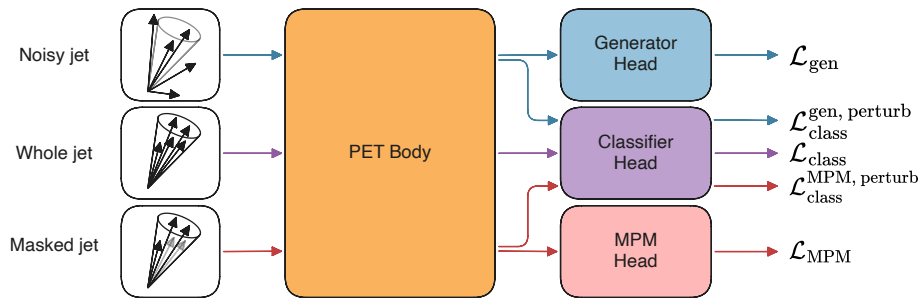


Figure 9.1: Illustration of the different jet augmentations and the forward passes performed for the different tasks. The model backbone is illustrated as “PET body” following the naming scheme of Ref. [214]. This figure was previously published in Ref. [5].

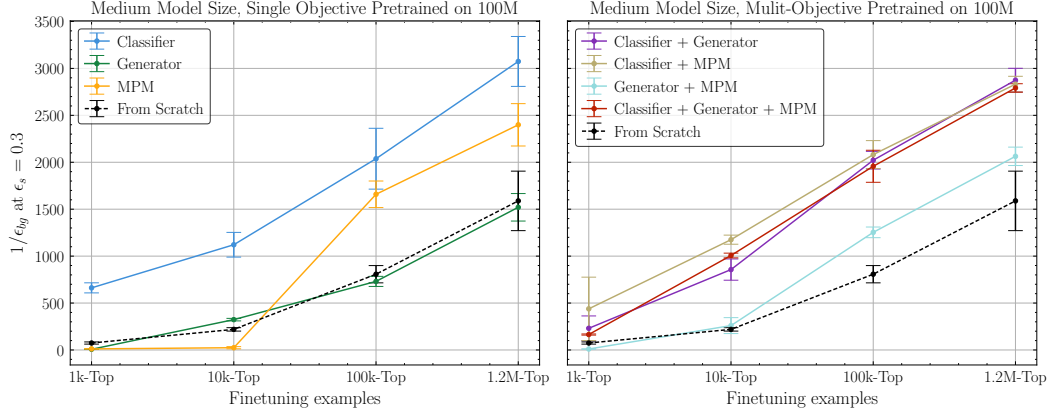


Figure 9.2: Downstream classification results with the medium model size as a function of the downstream dataset size. The left plot shows the results obtained with single-objective pre-training and the right plot those obtained with multi-objective pre-trainings. This figure was previously published in Ref. [5]. Uncertainties are estimated by reporting the mean and standard deviation across three classification trainings (with different random seeds) for each configuration.

section 4.3.3). During pre-training, the model uses features of all three feature categories (kinematics, PID, trajectory displacement) as input, similar to the model introduced in chapter 5. However, PID features as well as trajectory displacement information are passed through LayerDrop layers before entering the backbone. During training, these layers randomly drop those features with a fixed probability and set them to zeros such that the model is also trained for cases where this information is not available [210, 211]. The generative task and the MPM task are implemented such that they only predict the kinematic features of the jet constituents, i.e. the generative model generates the four kinematic features (p_T , $\Delta\eta$, $\Delta\phi$, E), and the MPM task regresses those features for the masked particles. The model backbone is based on the point-edge transformer (PET) architecture [210, 211, 214] which includes a local embedding block and a physics-informed attention bias [110], as well as registers [170]. The task-specific heads include smaller-sized transformer blocks that further process the backbone representation.

Three model sizes are investigated following the naming convention from Ref. [214]: micro (~ 100 k parameters), small (~ 7 M parameters), and medium (~ 51 M parameters). Downstream performance is studied as a function of the downstream dataset size. The classification task on the public top tagging benchmark dataset (see section 4.3.1) is used to evaluate downstream classification performance. The downstream dataset size N is varied over $N \in \{10^3, 10^4, 10^5, 1.2 \cdot 10^6\}$ and the background rejection $1/\epsilon_{\text{bkg}}$, evaluated at a signal efficiency of 30 %, is used to quantify the top tagging performance. That is, the decision boundary on the classifier output is chosen such that 30 % of the $t \rightarrow bq\bar{q}'$ jets have a top-jet probability larger than this boundary. Figure 9.2 shows the classification results of the fine-tuned models compared to the from-scratch baseline of the medium model size. The overall trend of the classification performance as a function of the downstream dataset size, and how it compares to the corresponding baseline, is found to be similar for the different model sizes [5]. The comparison of the single-objective pre-trainings (left plot in Figure 9.2) shows that pre-training on classification, i.e. on the same task as the downstream task except for it being a new dataset, leads to the best performance, with clear performance gains at all downstream dataset sizes. The two other single-objective pre-trainings behave very differently. Fine-tuning after pre-training with MPM leads to a background rejection worse

Table 9.1: Generative results of the models (small model size) trained on $N = 550$ k jets for a selection of the evaluated features. The values represent the Wasserstein distance W_1 between generated jets and jets from the JetNet dataset, averaged over all jet types. These results were previously published in Ref. [5].

	τ_{21} $\times 10^{-2}$	τ_{32} $\times 10^{-2}$	Jet energy (GeV)	Jet mass (GeV)
From Scratch	2.0 ± 0.3	2.1 ± 0.5	8.0 ± 1.2	0.9 ± 0.1
Generator	1.9 ± 0.3	2.0 ± 0.4	9.0 ± 2.3	1.2 ± 0.3
Classifier + Generator + MPM	1.2 ± 0.1	1.4 ± 0.4	8.1 ± 1.4	0.8 ± 0.2
Classifier + Generator	1.1 ± 0.2	1.4 ± 0.3	7.6 ± 1.2	0.7 ± 0.1
Generator + MPM	1.2 ± 0.2	1.4 ± 0.3	8.6 ± 1.3	0.9 ± 0.1
Classifier	3.6 ± 0.5	3.3 ± 0.6	9.2 ± 1.5	1.1 ± 0.2
MPM	3.0 ± 0.4	2.5 ± 0.6	9.2 ± 2.2	1.0 ± 0.2
Classifier + MPM	4.8 ± 0.8	4.0 ± 0.8	8.5 ± 1.4	1.2 ± 0.2

than the from-scratch baseline for $N = 1$ k and $N = 10$ k. For larger downstream dataset sizes, pre-training on MPM leads to a performance increase over the from-scratch baseline, but does not reach the background rejection achieved with classification pre-training. It should be noted that this trend is different for the small and micro model sizes, where the MPM-pre-trained models perform at least as well as the from-scratch baseline for all dataset sizes [5]. This indicates that larger models struggle to adapt from the representation learned during MPM pre-training to the classification task. Pre-training on the generative task alone does not result in a performance gain at any downstream dataset size. This is especially interesting as pre-training on jet generation with NTP (as opposed to flow matching) is found to lead to a performance gain in downstream classification [2, 4]. However, a direct comparison with the studies in chapters 6 and 8 is difficult, as those two works use different model architectures than the one here, and thus compare the fine-tuned models to different baseline models. A controlled study that compares the two generative pre-training approaches directly would be a valuable future study. Pre-training on multiple objectives at the same time (right plot in Figure 9.2) shows that the combined pre-trainings all result in comparable downstream classification performance, with the only exception being the combination Generator+MPM. The other multi-objective pre-training methods all contain the classification objective, which is presumably the dominating component for this trend. As for the Generator+MPM pre-training, the medium model benefits from this pre-training for dataset sizes larger than 100 k. These results might seem like the best way to pre-train a model for downstream classification is to simply stick with pure classification for pre-training as well. However, additional studies revealed that the downstream performance as a function of the *pre-training* steps is more stable when combining the classification objective with the generative objective [5].

To evaluate the generative downstream performance, the pre-trained models are fine-tuned to jets from the JetNet dataset (see section 4.3.2). Three downstream dataset sizes N are studied: 10 k, 100 k and 550 k. The quality of the generated jets is evaluated both on particle level and on jet level. The agreement of the generated jets with the target distribution is quantified by calculating the Wasserstein distance W_1 for a large set of features, with smaller W_1 values indicating better agreement between the distributions. These metrics are calculated for each jet type individually, and subsequently averaged. The micro and the medium model sizes are found to struggle with the generative task, compared to the small model [5]. In the case of the micro model size this is likely due to the model simply being

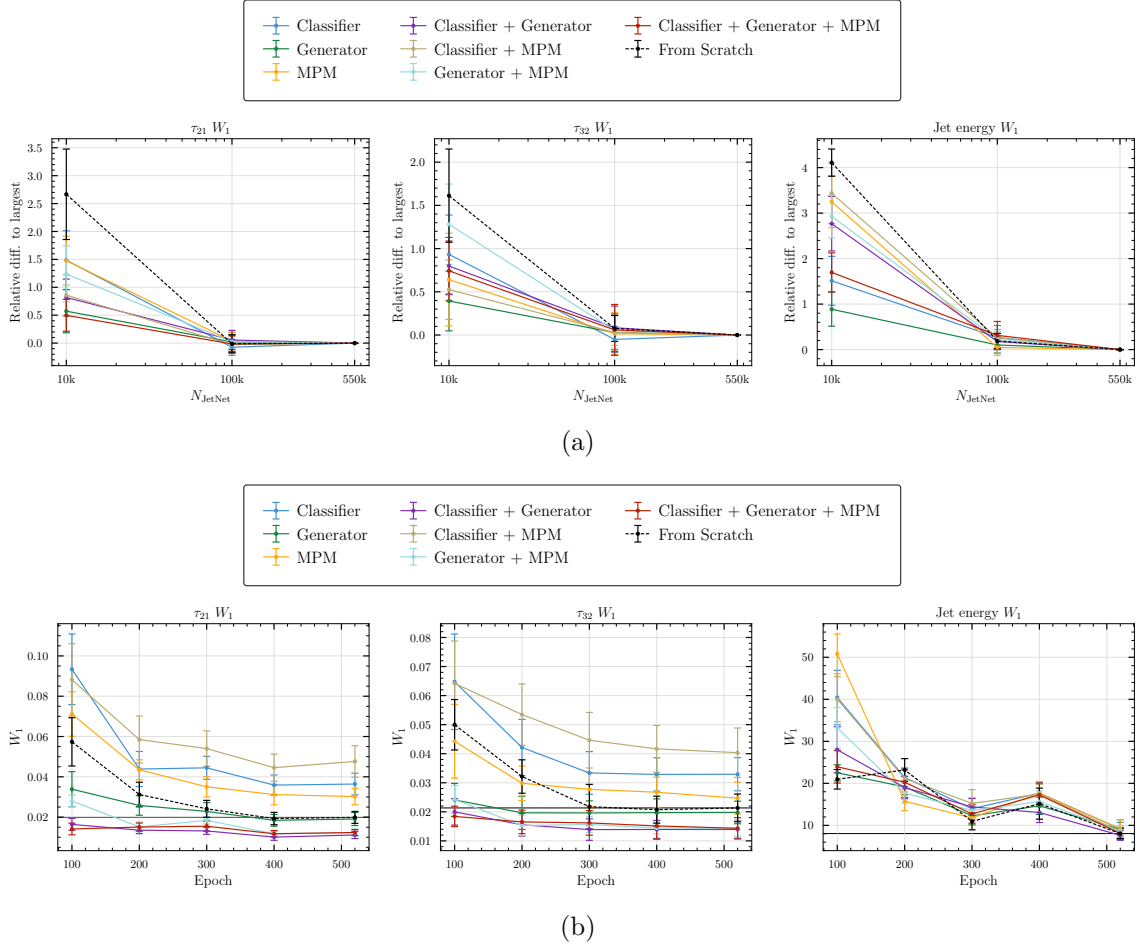


Figure 9.3: Generative performance of the small model. The upper panel (a) shows the relative performance as a function of the downstream dataset size N , denoted in the figure as N_{JetNet} . The performance is shown relative to the performance of the model trained with 550 k JetNet jets, i.e. the y -axis shows $(W_1(N) - W_1(550 \text{ k})) / W_1(550 \text{ k})$. The lower panel (b) shows the corresponding metrics as a function of the training epoch for the models trained with the largest dataset size (i.e. $N = 550 \text{ k}$). The horizontal line in (b) shows the final performance (at epoch 520) of the from-scratch baseline. Uncertainties are estimated with bootstrapping: the metrics are calculated five times, each time drawing 10 k jets from both the generated jets and the JetNet dataset.

too small for this task. That the medium model performs worse than the small model could be due to the fact that the medium-sized model might need a larger dataset to make use of the increased number of parameters compared to the small model size. This summary thus focuses on the small model size for the generative downstream task. The metrics of the generative model trained on 550 k JetNet jets, i.e. the largest downstream dataset size, are shown in Table 9.1. The best generative performance is achieved when the model is pre-trained at least partially on the generative task, as can be seen in the corresponding W_1 values in Table 9.1. The model pre-trained on the generative task alone performs on par with the from-scratch baseline whereas the models pre-trained jointly on generation and one or more other tasks perform better than the from-scratch model. Models that were pre-trained without the generative objective perform worse than the from-scratch baseline, indicating that those combinations of objectives are not well-aligned with the generative downstream

task.

Figure 9.3a shows the relative performance difference as a function of the downstream dataset size N with respect to the models trained with 550 k jets, thus illustrating how close to that performance the models with a smaller downstream dataset get. A substantial improvement is seen in the jet energy W_1 , with the relative improvement between the smallest and largest dataset size ranging from 100 % with generative pre-training to 400 % for the from-scratch baseline. The subjettiness ratio features also show significant improvements when comparing the smallest (10 k) and the largest (550 k) dataset size ranging from around 50 % to 250 %. From-scratch models benefit the most from the increased downstream dataset size, while the pre-trained models are closer to the corresponding $N = 550$ k performance already at smaller downstream dataset sizes.

In addition to the overall downstream performance and its dependency on the downstream dataset size, the convergence speed of the individual trainings is studied as well. This is visualized in Figure 9.3b, which shows the performance of the models trained with $N = 550$ k as a function of the training epoch. The trend in the convergence speed depends on the feature. For instance, the jet energy (right plot in Figure 9.3b) shows no clear improvement in convergence speed when pre-trained. Other features like the subjettiness ratios (left and middle plot in Figure 9.3b) show a clear improvement in convergence speed. Most notably, in τ_{21} and τ_{32} , the models pre-trained on the generative task combined with MPM, classification, or both, already reach better performance than the final (= epoch 520) from-scratch model after 100 or 200 training epochs.

These results provide a comprehensive study of how different supervised (classification) and unsupervised (generation and MPM) pre-training objectives as well as combinations of those affect the downstream performance in classification and generation. In downstream classification, the largest improvement is seen with pure classification pre-training, whereas combining classification with generative pre-training is found to lead to more stable pre-trained checkpoints. Pre-training on the flow-matching target alone, however, does not lead to a downstream performance gain in the classification task. Downstream generative performance, on the other hand, is found to only benefit from pre-training if the flow-matching objective is already included in the pre-training itself. Using a diverse combination of pre-training tasks therefore increases the range of downstream tasks that benefit from the pre-training as it reduces over-specialization of the pre-trained model, and can provide a balance between the individual, per-task achievable downstream performance gains and the breadth of different downstream tasks. Future studies will be needed to understand the apparent orthogonality of the generative downstream task and the classification task. Even more, future work should investigate the two different generative model paradigms, flow matching and NTP, and compare them in a controlled study.

10 Conclusion

The concept of foundation models [26] is a promising direction for machine learning applications in particle physics [198]. Pre-trained on large datasets, foundation models can substantially enhance downstream performance on a variety of tasks. To this end, appropriate pre-training methods need to be developed for foundation models in particle physics. While pre-training methods from other fields like NLP or computer vision can serve as useful inspiration, implementing those methods in particle physics is not straightforward. The field of particle physics comes with its own challenges but also with advantages. On the one hand, collider experiments record huge amounts of data, which makes simulation-free pre-training methods an attractive choice as they enable pre-training on measured data directly. On the other hand, the HEP community has developed tools that yield incredibly high-fidelity simulated data, which means that supervised pre-training methods are not categorically excluded as it is the case in many other fields where labeled datasets are scarce.

Over the last years, different pre-training methods were developed for particle physics, including masked modeling [222, 223, 226, 227], generative pre-training [2, 4], contrastive learning [228, 229, 231], self-distillation [235], and supervised approaches [110, 210, 211, 214, 236, 237]. This thesis presented multiple studies that contribute to the development of foundation models for particle physics, focusing on models that can generate and classify particle jets.

First, a flow-matching based generative model was developed [1]. This model is the first model that can generate not only the kinematic information of jet constituents, but also their particle-ID and trajectory displacement information.

The second study focuses on generative pre-training for jet foundation models [2]. This is realized by using the concept of next token prediction (NTP)-based pre-training on fully-tokenized particle jets, and investigates both jet generation and jet classification. The downstream classification task shows substantial performance gains when the model is pre-trained with NTP. Thus, this work demonstrates successful transfer learning between the jet generation task and jet classification, making this model, named OmniJet- α , the first cross-task foundation model for particle physics.

Given that NTP-based pre-training does not require external labels, it can be performed on measured data directly. This is studied in the third contribution [3] of this thesis. That work investigates the pre-training of a generative model with NTP on jets extracted from open data from the CMS experiment with subsequent fine-tuning to the generation of explicit jet types (q/g and $t \rightarrow bq\bar{q}'$) from a public benchmark dataset. Significant performance gains are observed when fine-tuning the pre-trained model with a small downstream dataset, thus demonstrating that pre-training on real collider data can lead to a downstream performance improvement. Additionally, this shows successful transfer learning within the jet generation task in a scenario with a large domain shift between the datasets used for pre-training and fine-tuning, respectively.

Next, multiple improvements are introduced to the original OmniJet- α -framework, addressing previous limitations of NTP-based pre-training for jet foundation models [4]. Instead of using the integer token-IDs obtained from tokenization, a hybrid setup is adapted. This hybrid setup uses continuous vectors as model input, while only keeping the token-IDs in the NTP target. As a result, the performance in the downstream classification task is significantly increased, while maintaining the generative capabilities attained from the generative pre-training. Furthermore, a joint pre-training method is introduced in this work, combining

NTP-based pre-training with the closely related MPM [222, 223]. A direct comparison of NTP, MPM and the joint NTP+MPM shows that pure MPM yields the best downstream classification, and clearly outperforms pre-training solely on NTP. The joint NTP+MPM pre-training method, however, offers a clear enhancement in downstream classification performance compared to pre-training on NTP alone. As opposed to NTP, MPM is just a proxy task for pre-training and does not yield a generative model. Simultaneously pre-training on both NTP and MPM does not harm the generative capabilities of the pre-trained model. This makes the combination NTP+MPM a powerful pre-training task for downstream classification while still benefitting from the generative nature of the NTP pre-training task.

The final contribution [5] presented a comparison study that evaluates different pre-training targets: classification, MPM, and jet generation via flow matching. This study thus provides a comparison of both supervised and unsupervised (= label-free) pre-training methods, and how those affect the performance of downstream jet classification and jet generation. Pre-training on classification was found to drastically boost downstream classification performance, outperforming both MPM and generative flow matching pre-training. In contrast, flow matching based pre-training alone is not found to improve downstream classification. However, similar to what was seen for NTP-based generative pre-training, combining the flow matching target with MPM improves downstream classification performance. No performance gain is seen in the generative downstream task, unless the generative task was already included during the pre-training. This shows the importance of aligning the pre-training task(s) and the downstream tasks.

As of now, it appears that combining multiple tasks is a promising way forward for foundation model pre-training in HEP. This yields significant performance gains in downstream tasks, while at the same time avoiding over-specialization during pre-training, thus offering an advantage to a range of downstream tasks. Different directions for future work arise from this work. First, existing pre-training methods should be further enhanced. As for NTP, this concerns more studies regarding high-resolution tokenization of jets. Tokenization can be challenging when extending the feature set [4]. Additionally, Ref. [277] found that binning-based tokenization of particle jets can lead to undesirable artifacts, and suggested that it should be investigated if those findings also affect VQ-VAE-based tokenization. Future studies in this direction would thus be valuable in order to verify the usability of generative models obtained from NTP pre-training with VQ-VAE-based tokenization.

Next, multiple generative pre-training paradigms for jet foundation models have been proposed thus far, including NTP [2, 4], diffusion [210, 211], flow matching [214], as well as set-to-set flow matching [223]. A thorough comparison of those methods, both in terms of their generative capabilities and their effect on other downstream tasks, would be valuable to navigate future efforts.

Finally, the development of foundation models for particle physics is still in its early stages, and was thus far mostly focused on proof-of-concept studies on public datasets. Future efforts should adapt methods developed in those proof-of-concept studies within experimental collaborations to fully exploit the potential of foundation models in particle physics.

Acknowledgements

First and foremost, I would like to thank my supervisor Gregor Kasieczka for his extensive support throughout my whole PhD. He always made time to discuss the projects I worked on, and I could rely on his guidance whenever needed. His ideas and advice were always of great help. I would also like to thank my second supervisor, Marcus Brüggem, for his support.

Very special thanks to Anna Hallin, Karim El Morabit, Dennis Noll, and Tanvi Wamorkar for their valuable feedback on the manuscript of this thesis.

I would like to thank all members of the local foundation models subgroup. The great discussions in our weekly meetings and the constant motivation to explore new approaches made this subgroup a great environment to work in. Here, I would like to especially thank Anna Hallin. I am very grateful for her constant support, her valuable advice, and the guidance she provided throughout the whole time of my PhD.

I am also grateful to all members of the larger Kasieczka group. Our group events, lunch breaks, and coffee breaks really brightened up the work routine.

Over the years, I had the pleasure of working with many great collaborators. I want to thank them for the great discussions and collaboration on the many projects we worked on together. Among them, I am particularly grateful to Ben Nachman for hosting me for a research stay at Stanford. I would also like to thank the German Academic Exchange Service (DAAD) for funding this research stay.

Looking back at my education as a physicist as a whole, I would also like to thank Andrea Knue and Gregor Herten. While only directly involved in my bachelor's and master's studies, they taught me many things that helped me during my PhD.

I would also like to thank my friends and family for their support over the years of my PhD.

Finally, I am especially grateful to Em, who has supported me through all the different stages of this PhD. Her tremendous support, care, and patience have been a fundamental pillar in this journey.

Thank you.

References

- [1] Joschka Birk, Erik Buhmann, Cedric Ewen, Gregor Kasieczka, and David Shih. “Flow matching beyond kinematics: Generating jets with particle identification and trajectory displacement information”. In: *Phys. Rev. D* 111.5 (2025), p. 052008. DOI: 10.1103/PhysRevD.111.052008. arXiv: 2312.00123 [hep-ph].
- [2] Joschka Birk, Anna Hallin, and Gregor Kasieczka. “OmniJet- α : the first cross-task foundation model for particle physics”. In: *Mach. Learn. Sci. Tech.* 5.3 (2024), p. 035031. DOI: 10.1088/2632-2153/ad66ad. arXiv: 2403.05618 [hep-ph].
- [3] Oz Amram, Luca Anzalone, Joschka Birk, Darius A. Faroughy, Anna Hallin, Gregor Kasieczka, Michael Krämer, Ian Pang, Humberto Reyes-Gonzalez, and David Shih. “Aspen Open Jets: unlocking LHC data for foundation models in particle physics”. In: *Mach. Learn. Sci. Tech.* 6.3 (2025), p. 030601. DOI: 10.1088/2632-2153/ade58f. arXiv: 2412.10504 [hep-ph].
- [4] Joschka Birk, Anna Hallin, Gregor Kasieczka, Nikol Madzharova, Ian Pang, and David Shih. “Enhancing next token prediction based pre-training for jet foundation models”. In: *Mach. Learn. Sci. Tech.* 7.3 (2026), p. 035042. DOI: 10.1088/2632-2153/ae70bc. arXiv: 2512.04149 [hep-ph].
- [5] Ibrahim Elsharkawy, Joschka Birk, Vinicius Mikuni, Wahid Bhimji, Gregor Kasieczka, and Benjamin Nachman. “Pre-Training for Simulation-Based Science: A Study on Jet Foundation Model Training Objectives” (June 2026). Submitted to *PRX Intelligence*, in review. arXiv: 2606.14870 [hep-ph].
- [6] Joschka Birk, Frank Gaede, Anna Hallin, Gregor Kasieczka, Martina Mozzanica, and Henning Rose. “OmniJet- α_C : learning point cloud calorimeter simulations using generative transformers”. In: *JINST* 20.07 (2025), P07007. DOI: 10.1088/1748-0221/20/07/P07007. arXiv: 2501.05534 [hep-ph].
- [7] Laurits Tani, Joosep Pata, and Joschka Birk. “Reconstructing hadronically decaying tau leptons with a jet foundation model”. In: *SciPost Phys. Core* 8 (2025), p. 046. DOI: 10.21468/SciPostPhysCore.8.3.046. arXiv: 2503.19165 [hep-ex].
- [8] Joschka Birk, Gregor Kasieczka, Siddharth Mishra-Sharma, Benjamin Nachman, Dennis Noll, and Tanvi Wamorkar. “A Scientific Human-Agent Reproduction Pipeline”. In: *Physics and AI Conference 2026 (PAI2026)*. June 2026. DOI: 10.5281/zenodo.21078068. arXiv: 2604.18752 [hep-ph].
- [9] Joschka Birk, Frank Gaede, Anna Hallin, Gregor Kasieczka, Martina Mozzanica, and Henning Rose. “SPADE: Split-and-Delay Embeddings for Autoregressive High-Granularity Calorimeter Simulation” (June 2026). Submitted to *Mach. Learn. Sci. Tech.* arXiv: 2606.11304 [physics.ins-det].
- [10] Dana Horwitz. “Integrating jet clustering history information in particle tagging”. Bachelor’s thesis. University of Hamburg, 2024.
- [11] Finn Johannsen. “Pre-training strategies for a foundation model in jet physics”. Master’s thesis. University of Hamburg, 2024.
- [12] Henning Rose. “OmniJet- α_C : Learning point cloud calorimeter simulations using generative transformers”. Master’s thesis. University of Hamburg, 2025.

-
- [13] Nikol Madzharova. “Improving Tokenisation for Foundation Models in Particle Physics”. Master’s thesis. University of Hamburg, 2026.
 - [14] The ATLAS Collaboration. “The ATLAS Experiment at the CERN Large Hadron Collider”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08003–S08003. DOI: 10.1088/1748-0221/3/08/s08003.
 - [15] The CMS Collaboration. “The CMS experiment at the CERN LHC”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08004–S08004. DOI: 10.1088/1748-0221/3/08/s08004.
 - [16] Lyndon Evans and Philip Bryant. “LHC Machine”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08001. DOI: 10.1088/1748-0221/3/08/S08001.
 - [17] O. Brüning, H. Burkhardt, and S. Myers. “The large hadron collider”. In: *Progress in Particle and Nuclear Physics* 67.3 (2012), pp. 705–734. ISSN: 0146-6410. DOI: 10.1016/j.pnpnp.2012.03.001.
 - [18] The ATLAS Collaboration. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 1–29. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.020.
 - [19] The CMS Collaboration. “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 30–61. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.021.
 - [20] P.W. Higgs. “Broken symmetries, massless particles and gauge fields”. In: *Physics Letters* 12.2 (1964), pp. 132–133. ISSN: 0031-9163. DOI: 10.1016/0031-9163(64)91136-9.
 - [21] Peter W. Higgs. “Broken Symmetries and the Masses of Gauge Bosons”. In: *Phys. Rev. Lett.* 13 (16 Oct. 1964), pp. 508–509. DOI: 10.1103/PhysRevLett.13.508.
 - [22] Peter W. Higgs. “Spontaneous Symmetry Breakdown without Massless Bosons”. In: *Phys. Rev.* 145 (4 May 1966), pp. 1156–1163. DOI: 10.1103/PhysRev.145.1156.
 - [23] F. Englert and R. Brout. “Broken Symmetry and the Mass of Gauge Vector Mesons”. In: *Phys. Rev. Lett.* 13 (9 Aug. 1964), pp. 321–323. DOI: 10.1103/PhysRevLett.13.321.
 - [24] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble. “Global Conservation Laws and Massless Particles”. In: *Phys. Rev. Lett.* 13 (20 Nov. 1964), pp. 585–587. DOI: 10.1103/PhysRevLett.13.585.
 - [25] T. W. B. Kibble. “Symmetry Breaking in Non-Abelian Gauge Theories”. In: *Phys. Rev.* 155 (5 Mar. 1967), pp. 1554–1561. DOI: 10.1103/PhysRev.155.1554.
 - [26] Rishi Bommasani et al. *On the Opportunities and Risks of Foundation Models*. 2021. arXiv: 2108.07258 [cs.LG].
 - [27] Christoph Berger and Gregor Herten. *Particle Physics: From the Basics to Modern Experiments*. Springer Berlin Heidelberg, 2025. ISBN: 9783662717066. DOI: 10.1007/978-3-662-71706-6.
 - [28] Cliff Burgess and Guy Moore. *The Standard Model - A Primer*. Cambridge: Cambridge University Press, 2007. ISBN: 978-0-521-86036-9.
 - [29] Michael E. Peskin and Daniel V. Schroeder. *An Introduction To Quantum Field Theory*. CRC Press, May 2018. DOI: 10.1201/9780429503559.
-

REFERENCES

- [30] David J Griffiths. *Introduction to elementary particles; 2nd rev. version*. Physics textbook. New York, NY: Wiley, 2008. URL: <https://cds.cern.ch/record/111880>.
- [31] Carsten Burgard. *Standard model of physics*. 2016. URL: <https://texample.net/tikz/examples/model-physics/>.
- [32] Particle Data Group Collaboration. “Review of Particle Physics”. In: *Physical Review D* 110.3 (Aug. 2024). ISSN: 2470-0029. DOI: 10.1103/physrevd.110.030001.
- [33] David J. Gross and Frank Wilczek. “Ultraviolet Behavior of Non-Abelian Gauge Theories”. In: *Physical Review Letters* 30.26 (June 1973), pp. 1343–1346. ISSN: 0031-9007. DOI: 10.1103/physrevlett.30.1343.
- [34] H. David Politzer. “Reliable Perturbative Results for Strong Interactions?” In: *Physical Review Letters* 30.26 (June 1973), pp. 1346–1349. ISSN: 0031-9007. DOI: 10.1103/physrevlett.30.1346.
- [35] Richard P. Feynman. *QED : the strange theory of light and matter*. eng. Harmondsworth: Penguin, 1990. ISBN: 0140125051.
- [36] Sheldon L. Glashow. “Partial-symmetries of weak interactions”. In: *Nuclear Physics* 22.4 (1961), pp. 579–588. ISSN: 0029-5582. DOI: 10.1016/0029-5582(61)90469-2.
- [37] Steven Weinberg. “A Model of Leptons”. In: *Phys. Rev. Lett.* 19 (21 Nov. 1967), pp. 1264–1266. DOI: 10.1103/PhysRevLett.19.1264.
- [38] Abdus Salam. “Gauge unification of fundamental forces”. In: *Rev. Mod. Phys.* 52 (3 July 1980), pp. 525–538. DOI: 10.1103/RevModPhys.52.525.
- [39] H. Fritzsch, M. Gell-Mann, and H. Leutwyler. “Advantages of the color octet gluon picture”. In: *Physics Letters B* 47.4 (Nov. 1973), pp. 365–368. ISSN: 0370-2693. DOI: 10.1016/0370-2693(73)90625-4.
- [40] Hideki Yukawa. “On the Interaction of Elementary Particles”. In: *Progress of Theoretical Physics Supplement* 1 (1955), pp. 1–10. ISSN: 0375-9687. DOI: 10.1143/ptps.1.1.
- [41] Howard Georgi and S. L. Glashow. “Unity of All Elementary-Particle Forces”. In: *Physical Review Letters* 32.8 (Feb. 1974), pp. 438–441. ISSN: 0031-9007. DOI: 10.1103/physrevlett.32.438.
- [42] Peter Minkowski. “ $\mu \rightarrow e\gamma$ at a rate of one out of 10^9 muon decays?” In: *Physics Letters B* 67.4 (Apr. 1977), pp. 421–428. ISSN: 0370-2693. DOI: 10.1016/0370-2693(77)90435-x.
- [43] Tsutomu Yanagida. “Horizontal gauge symmetry and masses of neutrinos”. In: *Conf. Proc. C* 7902131 (1979), pp. 95–99.
- [44] Rabindra N. Mohapatra and Goran Senjanovic. “Neutrino Mass and Spontaneous Parity Nonconservation”. In: *Phys. Rev. Lett.* 44 (1980), p. 912. DOI: 10.1103/PhysRevLett.44.912.
- [45] Murray Gell-Mann, Pierre Ramond, and Richard Slansky. “Complex Spinors and Unified Theories”. In: *Conf. Proc. C* 790927 (1979), pp. 315–321. arXiv: 1306.4669 [hep-th].
- [46] P. Ramond. *The Family Group in Grand Unified Theories*. 1998. arXiv: hep-ph/9809459 [hep-ph].
- [47] A. Einstein. “Die Grundlage der allgemeinen Relativitätstheorie”. In: *Annalen der Physik* 354.7 (Jan. 1916), pp. 769–822. ISSN: 1521-3889. DOI: 10.1002/andp.19163540702.

-
- [48] F. Zwicky. “Die Rotverschiebung von extragalaktischen Nebeln”. In: *Helv. Phys. Acta* 6 (1933). Republished in *Gen. Rel. Grav.* 41 (2009) 207–224, pp. 110–127. DOI: 10.1007/s10714-008-0707-4.
 - [49] Vera C. Rubin and W. Kent Ford Jr. “Rotation of the Andromeda Nebula from a Spectroscopic Survey of Emission Regions”. In: *Astrophys. J.* 159 (1970), pp. 379–403. DOI: 10.1086/150317.
 - [50] Vera C. Rubin, W. Kent Ford Jr., and Norbert Thonnard. “Extended rotation curves of high-luminosity spiral galaxies. IV. Systematic dynamical properties, Sa through Sc”. In: *Astrophys. J. Lett.* 225 (1978), pp. L107–L111. DOI: 10.1086/182804.
 - [51] D. Walsh, R. F. Carswell, and R. J. Weymann. “0957 + 561 A, B - Twin quasistellar objects or gravitational lens”. In: *Nature* 279 (1979), pp. 381–384. DOI: 10.1038/279381a0.
 - [52] Douglas Clowe, Marusa Bradac, Anthony H. Gonzalez, Maxim Markevitch, Scott W. Randall, Christine Jones, and Dennis Zaritsky. “A direct empirical proof of the existence of dark matter”. In: *Astrophys. J. Lett.* 648 (2006), pp. L109–L113. DOI: 10.1086/508162. arXiv: astro-ph/0608407.
 - [53] Scott Dodelson and Lawrence M. Widrow. “Sterile-neutrinos as dark matter”. In: *Phys. Rev. Lett.* 72 (1994), pp. 17–20. DOI: 10.1103/PhysRevLett.72.17. arXiv: hep-ph/9303287.
 - [54] R. D. Peccei and Helen R. Quinn. “Constraints Imposed by CP Conservation in the Presence of Instantons”. In: *Phys. Rev. D* 16 (1977), pp. 1791–1797. DOI: 10.1103/PhysRevD.16.1791.
 - [55] R. D. Peccei and Helen R. Quinn. “CP Conservation in the Presence of Instantons”. In: *Phys. Rev. Lett.* 38 (1977), pp. 1440–1443. DOI: 10.1103/PhysRevLett.38.1440.
 - [56] Steven Weinberg. “A New Light Boson?” In: *Phys. Rev. Lett.* 40 (1978), pp. 223–226. DOI: 10.1103/PhysRevLett.40.223.
 - [57] Frank Wilczek. “Problem of Strong P and T Invariance in the Presence of Instantons”. In: *Phys. Rev. Lett.* 40 (1978), pp. 279–282. DOI: 10.1103/PhysRevLett.40.279.
 - [58] H. Haber. “The search for supersymmetry: Probing physics beyond the standard model”. In: *Physics Reports* 117.2-4 (Jan. 1985), pp. 75–263. ISSN: 0370-1573. DOI: 10.1016/0370-1573(85)90051-1.
 - [59] Stephen P. Martin. “A Supersymmetry Primer”. In: *Perspectives on Supersymmetry*. WORLD SCIENTIFIC, July 1998, pp. 1–98. ISBN: 9789812839657. DOI: 10.1142/9789812839657_0001.
 - [60] Gary Steigman. “Observational Tests of Antimatter Cosmologies”. In: *Annual Review of Astronomy and Astrophysics* 14.1 (Sept. 1976), pp. 339–372. ISSN: 1545-4282. DOI: 10.1146/annurev.aa.14.090176.002011.
 - [61] Andrew G. Cohen, A. De Rujula, and S. L. Glashow. “A Matter - antimatter universe?” In: *Astrophys. J.* 495 (1998), pp. 539–549. DOI: 10.1086/305328. arXiv: astro-ph/9707087.
 - [62] Steven Weinberg. “Cosmological Production of Baryons”. In: *Phys. Rev. Lett.* 42 (1979), pp. 850–853. DOI: 10.1103/PhysRevLett.42.850.
 - [63] D. Toussaint, S. B. Treiman, Frank Wilczek, and A. Zee. “Matter - Antimatter Accounting, Thermodynamics, and Black Hole Radiation”. In: *Phys. Rev. D* 19 (1979), pp. 1036–1045. DOI: 10.1103/PhysRevD.19.1036.
-

REFERENCES

- [64] Ian Affleck and Michael Dine. “A New Mechanism for Baryogenesis”. In: *Nucl. Phys. B* 249 (1985), pp. 361–380. DOI: 10.1016/0550-3213(85)90021-5.
- [65] V. A. Kuzmin, V. A. Rubakov, and M. E. Shaposhnikov. “On the Anomalous Electroweak Baryon Number Nonconservation in the Early Universe”. In: *Phys. Lett. B* 155 (1985), p. 36. DOI: 10.1016/0370-2693(85)91028-7.
- [66] M. Fukugita and T. Yanagida. “Baryogenesis Without Grand Unification”. In: *Phys. Lett. B* 174 (1986), pp. 45–47. DOI: 10.1016/0370-2693(86)91126-3.
- [67] J. J. Thomson. “Cathode Rays”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 44.269 (Oct. 1897), pp. 293–316. ISSN: 1941-5990. DOI: 10.1080/14786449708621070.
- [68] A. Einstein. “Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt”. In: *Annalen der Physik* 322.6 (Jan. 1905), pp. 132–148. ISSN: 1521-3889. DOI: 10.1002/andp.19053220607.
- [69] S. H. Neddermeyer and C. D. Anderson. “Note on the Nature of Cosmic Ray Particles”. In: *Phys. Rev.* 51 (1937), pp. 884–886. DOI: 10.1103/PhysRev.51.884.
- [70] Frederick Reines and Clyde L. Cowan. “The neutrino”. In: *Nature* 178 (1956), pp. 446–449. DOI: 10.1038/178446a0.
- [71] G. Danby, J. M. Gaillard, Konstantin A. Goulianos, L. M. Lederman, Nari B. Mistry, M. Schwartz, and J. Steinberger. “Observation of High-Energy Neutrino Reactions and the Existence of Two Kinds of Neutrinos”. In: *Phys. Rev. Lett.* 9 (1962), pp. 36–44. DOI: 10.1103/PhysRevLett.9.36.
- [72] Elliott D. Bloom et al. “High-Energy Inelastic e-p Scattering at 6-Degrees and 10-Degrees”. In: *Phys. Rev. Lett.* 23 (1969), pp. 930–934. DOI: 10.1103/PhysRevLett.23.930.
- [73] Martin Breidenbach, Jerome I. Friedman, Henry W. Kendall, Elliott D. Bloom, D. H. Coward, H. C. DeStaebler, J. Drees, Luke W. Mo, and Richard E. Taylor. “Observed behavior of highly inelastic electron-proton scattering”. In: *Phys. Rev. Lett.* 23 (1969), pp. 935–939. DOI: 10.1103/PhysRevLett.23.935.
- [74] Martin L. Perl et al. “Evidence for Anomalous Lepton Production in e^+e^- Annihilation”. In: *Phys. Rev. Lett.* 35 (1975), pp. 1489–1492. DOI: 10.1103/PhysRevLett.35.1489.
- [75] G. Arnison et al. “Experimental Observation of Lepton Pairs of Invariant Mass Around $95\text{-GeV}/c^2$ at the CERN SPS Collider”. In: *Phys. Lett. B* 126 (1983), pp. 398–410. DOI: 10.1016/0370-2693(83)90188-0.
- [76] J. E. Augustin et al. “Discovery of a Narrow Resonance in e^+e^- Annihilation”. In: *Phys. Rev. Lett.* 33 (1974), pp. 1406–1408. DOI: 10.1103/PhysRevLett.33.1406.
- [77] J. J. Aubert et al. “Experimental Observation of a Heavy Particle J ”. In: *Phys. Rev. Lett.* 33 (1974), pp. 1404–1406. DOI: 10.1103/PhysRevLett.33.1404.
- [78] F. Abe et al. “Observation of top quark production in $\bar{p}p$ collisions”. In: *Phys. Rev. Lett.* 74 (1995), pp. 2626–2631. DOI: 10.1103/PhysRevLett.74.2626. arXiv: hep-ex/9503002.
- [79] S. Abachi et al. “Observation of the top quark”. In: *Phys. Rev. Lett.* 74 (1995), pp. 2632–2637. DOI: 10.1103/PhysRevLett.74.2632. arXiv: hep-ex/9503003.

-
- [80] The LHCb Collaboration. “The LHCb Detector at the LHC”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08005–S08005. DOI: 10.1088/1748-0221/3/08/s08005.
- [81] The ALICE Collaboration. “The ALICE experiment at the CERN LHC”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08002–S08002. DOI: 10.1088/1748-0221/3/08/s08002.
- [82] Izaak Neutelings. *The CMS coordinate system including the LHC*. 2021. URL: https://tikz.net/axis3d_cms/.
- [83] Izaak Neutelings. *Pseudorapidity*. 2021. URL: https://tikz.net/axis2d_pseudorapidity/.
- [84] A.M. Sirunyan et al. “Particle-flow reconstruction and global event description with the CMS detector”. In: *Journal of Instrumentation* 12.10 (Oct. 2017), P10003–P10003. ISSN: 1748-0221. DOI: 10.1088/1748-0221/12/10/p10003.
- [85] W. Adam et al. “The CMS high level trigger”. In: *Eur. Phys. J. C* 46 (2006), pp. 605–667. DOI: 10.1140/epjc/s2006-02495-8. arXiv: hep-ex/0512077.
- [86] Johannes Albrecht et al. “Summary of the trigger systems of the Large Hadron Collider experiments ALICE, ATLAS, CMS and LHCb”. In: *J. Phys. G* 52.3 (2025), p. 030501. DOI: 10.1088/1361-6471/adaadc. arXiv: 2408.03881 [physics.ins-det].
- [87] Dipak Maity. “CMS High Level Trigger(HLT) Performance in Run 3”. In: *Springer Proc. Phys.* 432 (2026), pp. 1513–1516. DOI: 10.1007/978-981-95-1513-4_346.
- [88] Georges Aad et al. “The ATLAS trigger system for LHC Run 3 and trigger performance in 2022”. In: *JINST* 19.06 (2024), P06029. DOI: 10.1088/1748-0221/19/06/P06029. arXiv: 2401.06630 [hep-ex].
- [89] Christian Bierlich et al. “A comprehensive guide to the physics and usage of PYTHIA 8.3”. In: *SciPost Phys. Codebases* (2022), p. 8. DOI: 10.21468/SciPostPhysCodeb.8. URL: <https://scipost.org/10.21468/SciPostPhysCodeb.8>.
- [90] Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. “A Brief Introduction to PYTHIA 8.1”. In: *Comput. Phys. Commun.* 178 (2008), pp. 852–867. DOI: 10.1016/j.cpc.2008.01.036. arXiv: 0710.3820 [hep-ph].
- [91] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands. “An introduction to PYTHIA 8.2”. In: *Computer Physics Communications* 191 (June 2015), pp. 159–177. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2015.01.024.
- [92] Johannes Bellm et al. “Herwig 7.0/Herwig++ 3.0 release note”. In: *The European Physical Journal C* 76.4 (Apr. 2016). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-016-4018-8.
- [93] Manuel Bähr et al. “Herwig++ physics and manual”. In: *The European Physical Journal C* 58.4 (Nov. 2008), pp. 639–707. DOI: 10.1140/epjc/s10052-008-0798-9.
- [94] T. Gleisberg, Stefan. Hoeche, F. Krauss, M. Schonherr, S. Schumann, F. Siegert, and J. Winter. “Event generation with SHERPA 1.1”. In: *JHEP* 02 (2009), p. 007. DOI: 10.1088/1126-6708/2009/02/007. arXiv: 0811.4622 [hep-ph].
- [95] Enrico Bothmann et al. “Event Generation with Sherpa 2.2”. In: *SciPost Phys.* 7.3 (2019), p. 034. DOI: 10.21468/SciPostPhys.7.3.034. arXiv: 1905.09127 [hep-ph].
-

- [96] Steffen Schumann and Frank Krauss. “A Parton shower algorithm based on Catani-Seymour dipole factorisation”. In: *JHEP* 03 (2008), p. 038. DOI: 10.1088/1126-6708/2008/03/038. arXiv: 0709.1027 [hep-ph].
- [97] Paolo Nason. “A New method for combining NLO QCD with shower Monte Carlo algorithms”. In: *JHEP* 11 (2004), p. 040. DOI: 10.1088/1126-6708/2004/11/040. arXiv: hep-ph/0409146.
- [98] Stefano Frixione, Paolo Nason, and Carlo Oleari. “Matching NLO QCD computations with Parton Shower simulations: the POWHEG method”. In: *JHEP* 11 (2007), p. 070. DOI: 10.1088/1126-6708/2007/11/070. arXiv: 0709.2092 [hep-ph].
- [99] Simone Alioli, Paolo Nason, Carlo Oleari, and Emanuele Re. “A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX”. In: *JHEP* 06 (2010), p. 043. DOI: 10.1007/JHEP06(2010)043. arXiv: 1002.2581 [hep-ph].
- [100] Heribertus B. Hartanto, Barbara Jager, Laura Reina, and Doreen Wackerroth. “Higgs boson production in association with top quarks in the POWHEG BOX”. In: *Phys. Rev. D* 91.9 (2015), p. 094003. DOI: 10.1103/PhysRevD.91.094003. arXiv: 1501.04498 [hep-ph].
- [101] Stefano Frixione, Paolo Nason, and Giovanni Ridolfi. “A Positive-weight next-to-leading-order Monte Carlo for heavy flavour hadroproduction”. In: *JHEP* 09 (2007), p. 126. DOI: 10.1088/1126-6708/2007/09/126. arXiv: 0707.3088 [hep-ph].
- [102] Johan Alwall, Pavel Demin, Simon de Visscher, Rikkert Frederix, Michel Herquet, Fabio Maltoni, Tilman Plehn, David L Rainwater, and Tim Stelzer. “MadGraph / MadEvent v4: the new web generation”. In: *Journal of High Energy Physics* 2007.09 (Sept. 2007), pp. 028–028. DOI: 10.1088/1126-6708/2007/09/028.
- [103] J.-C. Winter, F. Krauss, and G. Soff. “A modified cluster-hadronisation model”. In: *The European Physical Journal C* 36.3 (Aug. 2004), pp. 381–395. ISSN: 1434-6052. DOI: 10.1140/epjc/s2004-01960-8.
- [104] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjöstrand. “Parton fragmentation and string dynamics”. In: *Physics Reports* 97.2-3 (July 1983), pp. 31–145. ISSN: 0370-1573. DOI: 10.1016/0370-1573(83)90080-7.
- [105] S. Agostinelli et al. “Geant4—a simulation toolkit”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303. ISSN: 0168-9002. DOI: [https://doi.org/10.1016/S0168-9002\(03\)01368-8](https://doi.org/10.1016/S0168-9002(03)01368-8).
- [106] J. Allison et al. “Recent developments in Geant4”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 835 (Nov. 2016), pp. 186–225. ISSN: 0168-9002. DOI: 10.1016/j.nima.2016.06.125.
- [107] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. “DELPHES 3, A modular framework for fast simulation of a generic collider experiment”. In: *JHEP* 02 (2014), p. 057. DOI: 10.1007/JHEP02(2014)057. arXiv: 1307.6346 [hep-ex].
- [108] Anja Butter et al. “The Machine Learning landscape of top taggers”. In: *SciPost Phys.* 7 (2019), p. 014. DOI: 10.21468/SciPostPhys.7.1.014. arXiv: 1902.09914 [hep-ph].

-
- [109] Gregor Kasieczka, Tilman Plehn, Jennifer Thompson, and Michael Russel. *Top Quark Tagging Reference Dataset*. Version v0 (2018_03_27). Zenodo, Mar. 2019. DOI: 10.5281/zenodo.2603256.
 - [110] Huilin Qu, Congqiao Li, and Sitian Qian. “Particle Transformer for Jet Tagging”. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 18281–18292. arXiv: 2202.03772 [hep-ph].
 - [111] Huilin Qu, Congqiao Li, and Sitian Qian. *JetClass: A Large-Scale Dataset for Deep Learning in Jet Physics*. Version 1.0.0. Zenodo, June 2022. DOI: 10.5281/zenodo.6619768.
 - [112] G. Apollinari, O. Brüning, T. Nakamoto, and Lucio Rossi. “High Luminosity Large Hadron Collider HL-LHC”. In: *CERN Yellow Rep.* 5 (2015), pp. 1–19. DOI: 10.5170/CERN-2015-005.1. arXiv: 1705.08830 [physics.acc-ph].
 - [113] Johannes Albrecht et al. “A Roadmap for HEP Software and Computing R&D for the 2020s”. In: *Comput. Softw. Big Sci.* 3.1 (2019), p. 7. DOI: 10.1007/s41781-018-0018-8. arXiv: 1712.06982 [physics.comp-ph].
 - [114] A Boehnlein et al. *HL-LHC Software and Computing Review Panel, 2nd Report*. Tech. rep. Geneva: CERN, 2022. URL: <https://cds.cern.ch/record/2803119>.
 - [115] Klaus Rabbertz. *Jet Physics at the LHC: The Strong Force beyond the TeV Scale*. Vol. 268. Springer Tracts in Modern Physics. Berlin: Springer, 2017. ISBN: 9783319421155. DOI: 10.1007/978-3-319-42115-5.
 - [116] R. K. Ellis, W. J. Stirling, and B. R. Webber. *QCD and Collider Physics*. Cambridge University Press, Oct. 1996. ISBN: 9780521545891. DOI: 10.1017/cbo9780511628788.
 - [117] George F. Sterman and Steven Weinberg. “Jets from Quantum Chromodynamics”. In: *Phys. Rev. Lett.* 39 (1977), p. 1436. DOI: 10.1103/PhysRevLett.39.1436.
 - [118] John E. Huth et al. “Toward a Standardization of Jet Definitions”. In: *1990 DPF Summer Study on High-energy Physics: Research Directions for the Decade (Snowmass 90)*. Dec. 1990, pp. 0134–136.
 - [119] Gerald C. Blazey et al. “Run II jet physics”. In: *Physics at Run II: QCD and Weak Boson Physics Workshop: Final General Meeting*. May 2000, pp. 47–77. arXiv: hep-ex/0005012.
 - [120] S. D. Ellis, J. Huston, K. Hatakeyama, P. Loch, and M. Tonnesmann. “Jets in hadron-hadron collisions”. In: *Prog. Part. Nucl. Phys.* 60 (2008), pp. 484–551. DOI: 10.1016/j.pnpnp.2007.12.002. arXiv: 0712.2447 [hep-ph].
 - [121] C. Buttar et al. “Standard Model Handles and Candles Working Group: Tools and Jets Summary Report”. In: *5th Les Houches Workshop on Physics at TeV Colliders*. Mar. 2008, pp. 121–214. arXiv: 0803.0678 [hep-ph].
 - [122] Gavin P. Salam. “Towards Jetography”. In: *Eur. Phys. J. C* 67 (2010), pp. 637–686. DOI: 10.1140/epjc/s10052-010-1314-6. arXiv: 0906.1833 [hep-ph].
 - [123] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. “The anti- k_t jet clustering algorithm”. In: *JHEP* 04 (2008), p. 063. DOI: 10.1088/1126-6708/2008/04/063. arXiv: 0802.1189 [hep-ph].
 - [124] S. Catani, Yuri L. Dokshitzer, M. H. Seymour, and B. R. Webber. “Longitudinally invariant K_t clustering algorithms for hadron hadron collisions”. In: *Nucl. Phys. B* 406 (1993), pp. 187–224. DOI: 10.1016/0550-3213(93)90166-M.
-

REFERENCES

- [125] Stephen D. Ellis and Davison E. Soper. “Successive combination jet algorithm for hadron collisions”. In: *Phys. Rev. D* 48 (1993), pp. 3160–3166. DOI: 10.1103/PhysRevD.48.3160. arXiv: hep-ph/9305266.
- [126] Yuri L. Dokshitzer, G. D. Leder, S. Moretti, and B. R. Webber. “Better jet clustering algorithms”. In: *JHEP* 08 (1997), p. 001. DOI: 10.1088/1126-6708/1997/08/001. arXiv: hep-ph/9707323.
- [127] M. Wobisch and T. Wengler. “Hadronization corrections to jet cross-sections in deep inelastic scattering”. In: *Workshop on Monte Carlo Generators for HERA Physics (Plenary Starting Meeting)*. Apr. 1998, pp. 270–279. arXiv: hep-ph/9907280.
- [128] Matteo Cacciari. “FastJet: A Code for fast k_t clustering, and more”. In: *41st Rencontres de Moriond: QCD and Hadronic Interactions*. July 2006, pp. 487–490. arXiv: hep-ph/0607071.
- [129] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. “FastJet User Manual”. In: *Eur. Phys. J. C* 72 (2012), p. 1896. DOI: 10.1140/epjc/s10052-012-1896-2. arXiv: 1111.6097 [hep-ph].
- [130] Jonathan M. Butterworth, Adam R. Davison, Mathieu Rubin, and Gavin P. Salam. “Jet substructure as a new Higgs search channel at the LHC”. In: *Phys. Rev. Lett.* 100 (2008), p. 242001. DOI: 10.1103/PhysRevLett.100.242001. arXiv: 0802.2470 [hep-ph].
- [131] A. Abdesselam et al. “Boosted Objects: A Probe of Beyond the Standard Model Physics”. In: *Eur. Phys. J. C* 71 (2011), p. 1661. DOI: 10.1140/epjc/s10052-011-1661-y. arXiv: 1012.5412 [hep-ph].
- [132] G Brooijmans. *High p_T Hadronic Top Quark Identification*. Tech. rep. Geneva: CERN, 2008. URL: <https://cds.cern.ch/record/1077731>.
- [133] Jesse Thaler and Lian-Tao Wang. “Strategies to Identify Boosted Tops”. In: *JHEP* 07 (2008), p. 092. DOI: 10.1088/1126-6708/2008/07/092. arXiv: 0806.0023 [hep-ph].
- [134] Leandro G. Almeida, Seung J. Lee, Gilad Perez, George F. Sterman, Ilmo Sung, and Joseph Virzi. “Substructure of high- p_T Jets at the LHC”. In: *Phys. Rev. D* 79 (2009), p. 074017. DOI: 10.1103/PhysRevD.79.074017. arXiv: 0807.0234 [hep-ph].
- [135] Jonathan M. Butterworth, Adam R. Davison, Mathieu Rubin, and Gavin P. Salam. “Jet substructure as a new Higgs search channel at the LHC”. In: *34th International Conference on High Energy Physics*. Oct. 2008. arXiv: 0810.0409 [hep-ph].
- [136] Stephen D. Ellis, Christopher K. Vermilion, and Jonathan R. Walsh. “Techniques for improved heavy particle searches with jet substructure”. In: *Phys. Rev. D* 80 (2009), p. 051501. DOI: 10.1103/PhysRevD.80.051501. arXiv: 0903.5081 [hep-ph].
- [137] Stephen D. Ellis, Christopher K. Vermilion, and Jonathan R. Walsh. “Recombination Algorithms and Jet Substructure: Pruning as a Tool for Heavy Particle Searches”. In: *Phys. Rev. D* 81 (2010), p. 094023. DOI: 10.1103/PhysRevD.81.094023. arXiv: 0912.0033 [hep-ph].
- [138] David Krohn, Jesse Thaler, and Lian-Tao Wang. “Jet Trimming”. In: *JHEP* 02 (2010), p. 084. DOI: 10.1007/JHEP02(2010)084. arXiv: 0912.1342 [hep-ph].
- [139] Davison E. Soper and Michael Spannowsky. “Combining subjet algorithms to enhance ZH detection at the LHC”. In: *JHEP* 08 (2010), p. 029. DOI: 10.1007/JHEP08(2010)029. arXiv: 1005.0417 [hep-ph].

-
- [140] Jesse Thaler and Ken Van Tilburg. “Identifying Boosted Objects with N-subjettiness”. In: *JHEP* 03 (2011), p. 015. DOI: 10.1007/JHEP03(2011)015. arXiv: 1011.2268 [hep-ph].
 - [141] Kim Albertsson et al. “Machine Learning in High Energy Physics Community White Paper”. In: *J. Phys. Conf. Ser.* 1085.2 (2018), p. 022008. DOI: 10.1088/1742-6596/1085/2/022008. arXiv: 1807.02876 [physics.comp-ph].
 - [142] Alexander Radovic, Mike Williams, David Rousseau, Michael Kagan, Daniele Bonacorsi, Alexander Himmel, Adam Aurisano, Kazuhiro Terao, and Taritree Wongjirad. “Machine learning at the energy and intensity frontiers of particle physics”. In: *Nature* 560.7716 (2018), pp. 41–48. DOI: 10.1038/s41586-018-0361-2.
 - [143] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naf-tali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. “Machine learning and the physical sciences”. In: *Rev. Mod. Phys.* 91.4 (2019), p. 045002. DOI: 10.1103/RevModPhys.91.045002. arXiv: 1903.10563 [physics.comp-ph].
 - [144] Dimitri Bourilkov. “Machine and Deep Learning Applications in Particle Physics”. In: *Int. J. Mod. Phys. A* 34.35 (2020), p. 1930019. DOI: 10.1142/S0217751X19300199. arXiv: 1912.08245 [physics.data-an].
 - [145] Matthew D. Schwartz. “Modern Machine Learning and Particle Physics” (Mar. 2021). DOI: 10.1162/99608f92.beeb1183. arXiv: 2103.12226 [hep-ph].
 - [146] Georgia Karagiorgi, Gregor Kasieczka, Scott Kravitz, Benjamin Nachman, and David Shih. “Machine Learning in the Search for New Fundamental Physics” (Dec. 2021). arXiv: 2112.03769 [hep-ph].
 - [147] Phiala Shanahan et al. “Snowmass 2021 Computational Frontier CompF03 Topical Group Report: Machine Learning” (2022). arXiv: 2209.07559 [physics.comp-ph].
 - [148] Thea Klaebye Aarrestad et al. “Building an AI-native Research Ecosystem for Experimental Particle Physics: A Community Vision” (Feb. 2026). arXiv: 2602.17582 [hep-ex].
 - [149] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023. URL: <http://udlbook.com>.
 - [150] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
 - [151] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].
 - [152] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations*. 2019. arXiv: 1711.05101 [cs.LG].
 - [153] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. “On the Variance of the Adaptive Learning Rate and Beyond”. In: *International Conference on Learning Representations*. 2020. arXiv: 1908.03265 [cs.LG].
 - [154] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. “Lookahead Optimizer: k steps forward, 1 step back”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. arXiv: 1907.08610 [cs.LG].
 - [155] Less Wright. *Ranger - a synergistic optimizer*. <https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer>. 2019.
-

REFERENCES

- [156] Izaak Neutelings. *Neural Networks*. 2021. URL: https://tikz.net/neural_networks/.
- [157] Seppo Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. Master’s Thesis (in Finnish). University of Helsinki, 1970.
- [158] Paul J. Werbos. “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences”. PhD Thesis. Harvard University, 1974.
- [159] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*. Technical Report. La Jolla Institute for Cognitive Science, UCSD, 1985.
- [160] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0.
- [161] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. arXiv: 1912.01703 [cs.LG].
- [162] Manzil Zaheer et al. “Deep Sets”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. arXiv: 1703.06114 [cs.LG].
- [163] Erik Buhmann, Gregor Kasieczka, and Jesse Thaler. “EPiC-GAN: Equivariant point cloud generation for particle jets”. In: *SciPost Phys.* 15.4 (2023), p. 130. DOI: 10.21468/SciPostPhys.15.4.130. arXiv: 2301.08128 [hep-ph].
- [164] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964. arXiv: 1706.03762 [cs.CL].
- [165] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. “PCT: Point cloud transformer”. In: *Computational Visual Media* 7.2 (2021), pp. 187–199. DOI: 10.1007/s41095-021-0229-5. arXiv: 2012.09688 [cs.CV].
- [166] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. *Point Transformer*. 2021. arXiv: 2012.09164 [cs.CV]. URL: <https://arxiv.org/abs/2012.09164>.
- [167] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.
- [168] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].
- [169] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. “On layer normalization in the transformer architecture”. In: *Proceedings of the 37th International Conference on Machine Learning*. ICML’20. JMLR.org, 2020. arXiv: 2002.04745 [cs.LG].
- [170] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. “Vision Transformers Need Registers”. In: *The Twelfth International Conference on Learning Representations*. 2024. arXiv: 2309.16588 [cs.CV].

-
- [171] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. “Going Deeper With Image Transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 32–42. arXiv: 2103.17239 [cs.CV].
 - [172] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. “Flow Matching for Generative Modeling”. In: *The Eleventh International Conference on Learning Representations*. 2023. arXiv: 2210.02747 [cs.LG].
 - [173] Michael Samuel Albergo and Eric Vanden-Eijnden. “Building Normalizing Flows with Stochastic Interpolants”. In: *The Eleventh International Conference on Learning Representations*. 2023. arXiv: 2209.15571 [cs.LG].
 - [174] Xingchao Liu, Chengyue Gong, and Qiang Liu. “Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow”. In: *The Eleventh International Conference on Learning Representations*. 2023. arXiv: 2209.03003 [cs.LG].
 - [175] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. “Neural Ordinary Differential Equations”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. arXiv: 1806.07366 [cs.LG].
 - [176] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. *Flow Matching Guide and Code*. 2024. arXiv: 2412.06264 [cs.LG].
 - [177] Claude Elwood Shannon. “A Mathematical Theory of Communication”. In: *The Bell System Technical Journal* 27 (1948), pp. 379–423.
 - [178] C. E. Shannon. “Prediction and Entropy of Printed English”. In: *Bell System Technical Journal* 30.1 (Jan. 1951), pp. 50–64. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1951.tb01366.x.
 - [179] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. *Improving language understanding by generative pre-training*. 2018.
 - [180] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. 2019.
 - [181] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].
 - [182] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.
 - [183] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”. In: *J. Mach. Learn. Res.* 11 (Dec. 2010), pp. 3371–3408. ISSN: 1532-4435.
 - [184] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. “Context Encoders: Feature Learning by Inpainting”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016. arXiv: 1604.07379 [cs.CV].
 - [185] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. “BEiT: BERT Pre-Training of Image Transformers”. In: *International Conference on Learning Representations*. 2022. arXiv: 2106.08254 [cs.CV].
-

REFERENCES

- [186] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. “Masked Autoencoders Are Scalable Vision Learners”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 16000–16009. arXiv: 2111.06377 [cs.CV].
- [187] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. “Masked autoencoders for point cloud self-supervised learning”. In: *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*. Springer. 2022, pp. 604–621. arXiv: 2203.06604 [cs.CV].
- [188] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. 2021. arXiv: 2010.11929 [cs.CV].
- [189] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. “Neural Discrete Representation Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. arXiv: 1711.00937 [cs.LG].
- [190] Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. “Straightening Out the Straight-Through Estimator: Overcoming Optimization Challenges in Vector Quantized Networks”. In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. PMLR, July 2023, pp. 14096–14113. arXiv: 2305.08842 [cs.LG].
- [191] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. “Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation”. In: *CoRR* abs/1308.3432 (2013). arXiv: 1308.3432.
- [192] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 8748–8763. arXiv: 2103.00020 [cs.CV].
- [193] Sebastian Thrun. “Lifelong learning algorithms”. In: *Learning to Learn*. USA: Kluwer Academic Publishers, 1998, pp. 181–209. ISBN: 0792380479.
- [194] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG]. URL: <https://arxiv.org/abs/1411.1792>.
- [195] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG].
- [196] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. *Scaling Laws for Transfer*. 2021. arXiv: 2102.01293 [cs.LG].
- [197] Jordan Hoffmann et al. “An empirical analysis of compute-optimal large language model training”. In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 30016–30030. DOI: 10.52202/068431–2176.
- [198] Anna Hallin. “Foundation models for high-energy physics”. In: *2nd European AI for Fundamental Physics Conference*. Sept. 2025. arXiv: 2509.21434 [hep-ph].
- [199] HEP ML Community. *A Living Review of Machine Learning for Particle Physics*. URL: <https://iml-wg.github.io/HEPML-LivingReview/>.
- [200] G. Aad et al. “Software and computing for Run 3 of the ATLAS experiment at the LHC”. In: *Eur. Phys. J. C* 85.3 (2025), p. 234. DOI: 10.1140/epjc/s10052-024-13701-w. arXiv: 2404.06335 [hep-ex].

-
- [201] Georges Aad et al. “Transforming jet flavour tagging at ATLAS”. In: *Nature Commun.* 17.1 (2026), p. 541. DOI: 10.1038/s41467-025-65059-6. arXiv: 2505.19689 [hep-ex].
 - [202] *GN3: Multi-task, Multi-modal Transformers for Jet Flavour Tagging in ATLAS*. Tech. rep. Geneva: CERN, 2026. URL: <https://cds.cern.ch/record/2953652>.
 - [203] *Carpe Datum: Scaling behavior of transformers for heavy hadron flavor identification*. Tech. rep. Geneva: CERN, 2026. URL: <https://cds.cern.ch/record/2953659>.
 - [204] *Flavour tagging performance of the updated Unified Particle Transformer algorithm with the CMS experiment at $\sqrt{s}=13.6$ TeV*. Tech. rep. 2025. URL: <https://cds.cern.ch/record/2948917>.
 - [205] “A unified approach for jet tagging in Run 3 at $\sqrt{s}=13.6$ TeV in CMS” (2024). URL: <https://cds.cern.ch/record/2904702>.
 - [206] Georges Aad et al. “New techniques for jet calibration with the ATLAS detector”. In: *Eur. Phys. J. C* 83.8 (2023), p. 761. DOI: 10.1140/epjc/s10052-023-11837-9. arXiv: 2303.17312 [hep-ex].
 - [207] Vardan Khachatryan et al. “Jet energy scale and resolution in the CMS experiment in pp collisions at 8 TeV”. In: *JINST* 12.02 (2017), P02014. DOI: 10.1088/1748-0221/12/02/P02014. arXiv: 1607.03663 [hep-ex].
 - [208] Georges Aad et al. “Track and Vertex Reconstruction with the ATLAS Inner Detector” (May 2026). arXiv: 2605.07585 [physics.ins-det].
 - [209] Erik Buhmann, Cedric Ewen, Gregor Kasieczka, Vinicius Mikuni, Benjamin Nachman, and David Shih. “Full phase space resonant anomaly detection”. In: *Phys. Rev. D* 109.5 (2024), p. 055015. DOI: 10.1103/PhysRevD.109.055015. arXiv: 2310.06897 [hep-ph].
 - [210] Vinicius Mikuni and Benjamin Nachman. “Solving key challenges in collider physics with foundation models”. In: *Phys. Rev. D* 111.5 (2025), p. L051504. DOI: 10.1103/PhysRevD.111.L051504. arXiv: 2404.16091 [hep-ph].
 - [211] Vinicius Mikuni and Benjamin Nachman. “A method to simultaneously facilitate all jet physics tasks”. In: *Phys. Rev. D* 111.5 (2025), p. 054015. DOI: 10.1103/PhysRevD.111.054015. arXiv: 2502.14652 [hep-ph].
 - [212] Vinicius Mikuni and Benjamin Nachman. “Searching for Anomalies with Foundation Models” (Mar. 2026). arXiv: 2603.23593 [hep-ex].
 - [213] Anna Hallin, Joshua Isaacson, Gregor Kasieczka, Claudius Krause, Benjamin Nachman, Tobias Quadfasel, Matthias Schlaffer, David Shih, and Manuel Sommerhalder. “Classifying anomalies through outer density estimation”. In: *Phys. Rev. D* 106.5 (2022), p. 055006. DOI: 10.1103/PhysRevD.106.055006. arXiv: 2109.00546 [hep-ph].
 - [214] Wahid Bhimji, Chris Harris, Vinicius Mikuni, and Benjamin Nachman. “Foundation model framework for all tasks involving jet physics”. In: *Phys. Rev. D* 113.3 (2026), p. 032020. DOI: 10.1103/knmd-f5jm. arXiv: 2510.24066 [hep-ph].
 - [215] Anja Butter, Gregor Kasieczka, Tilman Plehn, and Michael Russell. “Deep-learned Top Tagging with a Lorentz Layer”. In: *SciPost Phys.* 5.3 (2018), p. 028. DOI: 10.21468/SciPostPhys.5.3.028. arXiv: 1707.08966 [hep-ph].
 - [216] Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos. *JetNet150*. Version 2.0.0. Zenodo, Aug. 2022. DOI: 10.5281/zenodo.6975117.
-

REFERENCES

- [217] Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos. “Particle Cloud Generation with Message Passing Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 23858–23871. arXiv: 2106.11535 [cs.LG].
- [218] Maurizio Pierini, Javier Mauricio Duarte, Nhan Tran, and Marat Freytsis. *HLS4ML LHC Jet dataset (150 particles)*. Zenodo, Jan. 2020. DOI: 10.5281/zenodo.3602260. URL: <https://doi.org/10.5281/zenodo.3602260>.
- [219] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. -S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. “The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations”. In: *JHEP* 07 (2014), p. 079. DOI: 10.1007/JHEP07(2014)079. arXiv: 1405.0301 [hep-ph].
- [220] Erik Buhmann, Cedric Ewen, Darius A. Faroughy, Tobias Golling, Gregor Kasieczka, Matthew Leigh, Guillaume Quétant, John Andrew Raine, Debajyoti Sengupta, and David Shih. “EPiC-ly Fast Particle Cloud Generation with Flow-Matching and Diffusion” (Sept. 2023). arXiv: 2310.00049 [hep-ph].
- [221] Albert M Sirunyan et al. “Identification of heavy, energetic, hadronically decaying particles using machine-learning techniques”. In: *JINST* 15.06 (2020), P06005. DOI: 10.1088/1748-0221/15/06/P06005. arXiv: 2004.08262 [hep-ex].
- [222] Tobias Golling, Lukas Heinrich, Michael Kagan, Samuel Klein, Matthew Leigh, Margarita Osadchy, and John Andrew Raine. “Masked particle modeling on sets: towards self-supervised high energy physics foundation models”. In: *Mach. Learn. Sci. Tech.* 5.3 (2024), p. 035074. DOI: 10.1088/2632-2153/ad64a8. arXiv: 2401.13537 [hep-ph].
- [223] Matthew Leigh, Samuel Klein, François Charton, Tobias Golling, Lukas Heinrich, Michael Kagan, Inês Ochoa, and Margarita Osadchy. “Is Tokenization Needed for Masked Particle Modelling?” In: *Machine Learning: Science and Technology* 6.2 (June 2025), p. 025075. DOI: 10.1088/2632-2153/addb98. arXiv: 2409.12589 [hep-ph].
- [224] Yann LeCun. “A Path Towards Autonomous Machine Intelligence” (June 2022).
- [225] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. “Self-Supervised Learning From Images With a Joint-Embedding Predictive Architecture”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 15619–15629. arXiv: 2301.08243 [cs.CV].
- [226] Subash Katel, Haoyang Li, Zihan Zhao, Farouk Mokhtar, Javier Duarte, and Raghav Kansal. “Learning Symmetry-Independent Jet Representations via Jet-Based Joint Embedding Predictive Architecture”. In: *Machine Learning and the Physical Sciences: Workshop at NeurIPS 2024*. Dec. 2024. arXiv: 2412.05333 [hep-ph].
- [227] Jai Bardhan, Radhikesh Agrawal, Abhiram Tilak, Cyrin Neeraj, and Subhadip Mitra. “HEP-JEPA: A foundation model for collider physics using joint embedding predictive architecture” (Feb. 2025). arXiv: 2502.03933 [cs.LG].
- [228] Barry M. Dillon, Gregor Kasieczka, Hans Olschlager, Tilman Plehn, Peter Sorrenson, and Lorenz Vogel. “Symmetries, safety, and self-supervision”. In: *SciPost Phys.* 12.6 (2022), p. 188. DOI: 10.21468/SciPostPhys.12.6.188. arXiv: 2108.04253 [hep-ph].

-
- [229] Philip Harris, Jeffrey Krupa, Michael Kagan, Benedikt Maier, and Nathaniel Woodward. “Resimulation-based self-supervised learning for pretraining physics foundation models”. In: *Phys. Rev. D* 111.3 (2025), p. 032010. DOI: 10.1103/PhysRevD.111.032010. arXiv: 2403.07066 [hep-ph].
- [230] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 1597–1607. arXiv: 2002.05709 [cs.LG].
- [231] Zihan Zhao, Farouk Mokhtar, Raghav Kansal, Haoyang Li, and Javier Duarte. “Large-Scale Pretraining and Finetuning for Efficient Jet Classification in Particle Physics”. In: *22nd International Workshop on Advanced Computing and Analysis Techniques in Physics Research: Foundation Models for Physics - Nexus of Computation and Physics through Embracing the Era of Foundation Models*. Aug. 2024. arXiv: 2408.09343 [hep-ex].
- [232] Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. “Emerging Properties in Self-Supervised Vision Transformers”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021, pp. 9630–9640. DOI: 10.1109/iccv48922.2021.00951.
- [233] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2024. arXiv: 2304.07193 [cs.CV].
- [234] Oriane Siméoni et al. *DINOv3*. 2025. arXiv: 2508.10104 [cs.CV].
- [235] Zichun Hao, Raghav Kansal, Abhijith Gandrakota, Chang Sun, Jennifer Ngadiuba, Javier Duarte, and Maria Spiropulu. “RINO: Renormalization Group Invariance with No Labels”. In: *Machine Learning and the Physical Sciences Workshop @ NeurIPS 2025 (Spotlight)* (2025). arXiv: 2509.07486.
- [236] Jonas Spinner, Victor Bresó, Pim De Haan, Tilman Plehn, Jesse Thaler, and Johann Brehmer. “Lorentz-Equivariant Geometric Algebra Transformers for High-Energy Physics”. In: *Advances in Neural Information Processing Systems*. Vol. 37. 2024. arXiv: 2405.14806 [physics.data-an].
- [237] Johann Brehmer, Víctor Bresó, Pim de Haan, Tilman Plehn, Huilin Qu, Jonas Spinner, and Jesse Thaler. “A Lorentz-equivariant transformer for all of the LHC”. In: *SciPost Phys.* 19.4 (2025), p. 108. DOI: 10.21468/SciPostPhys.19.4.108. arXiv: 2411.00446 [hep-ph].
- [238] Congqiao Li et al. “Accelerating Resonance Searches via Signature-Oriented Pre-training” (May 2024). arXiv: 2405.12972 [hep-ph].
- [239] Matthias Vigl, Nicole Hartman, and Lukas Heinrich. “Finetuning foundation models for joint analysis optimization in High Energy Physics”. In: *Mach. Learn. Sci. Tech.* 5.2 (2024), p. 025075. DOI: 10.1088/2632-2153/ad55a3. arXiv: 2401.13536 [hep-ex].
- [240] Farouk Mokhtar, Joosep Pata, Dolores Garcia, Eric Wulff, Mengke Zhang, Michael Kagan, and Javier Duarte. “Fine-tuning machine-learned particle-flow reconstruction for new detector geometries in future colliders”. In: *Phys. Rev. D* 111.9 (2025), p. 092015. DOI: 10.1103/PhysRevD.111.092015. arXiv: 2503.00131 [hep-ex].
-

REFERENCES

- [241] Andrew J. Wildridge, Jack P. Rodgers, Ethan M. Colbert, Yao Yao, Andreas W. Jung, and Miaoyuan Liu. “Bumblebee: Foundation Model for Particle Physics Discovery”. In: *Machine Learning and the Physical Sciences: Workshop at NeurIPS 2024*. Dec. 2024. arXiv: 2412.07867 [hep-ex].
- [242] Joshua Ho, Benjamin Ryan Roberts, Shuo Han, and Haichen Wang. “Pretrained Event Classification Model for High Energy Physics Analysis” (Dec. 2024). arXiv: 2412.10665 [hep-ph].
- [243] Raghav Kansal, Anni Li, Javier Duarte, Nadezda Chernyavskaya, Maurizio Pierini, Breno Orzari, and Thiago Tomei. “Evaluating generative models in high energy physics”. In: *Phys. Rev. D* 107.7 (2023), p. 076017. DOI: 10.1103/PhysRevD.107.076017. arXiv: 2211.10295 [hep-ex].
- [244] Benno Käch and Isabell Melzer-Pellmann. “Attention to Mean-Fields for Particle Cloud Generation” (May 2023). arXiv: 2305.15254 [hep-ex].
- [245] Anni Li, Venkat Krishnamohan, Raghav Kansal, Rounak Sen, Steven Tsan, Zhaoyu Zhang, and Javier Duarte. “Induced Generative Adversarial Particle Transformers”. In: *37th Conference on Neural Information Processing Systems*. Dec. 2023. arXiv: 2312.04757 [hep-ex].
- [246] Benno Käch, Dirk Krücker, Isabell Melzer-Pellmann, Moritz Scham, Simon Schnake, and Alexi Verney-Provatas. “JetFlow: Generating Jets with Conditioned and Mass Constrained Normalising Flows” (Nov. 2022). arXiv: 2211.13630 [hep-ex].
- [247] Matthew Leigh, Debajyoti Sengupta, Guillaume Quétant, John Andrew Raine, Knut Zoch, and Tobias Golling. “PC-JeDi: Diffusion for particle cloud generation in high energy physics”. In: *SciPost Phys.* 16.1 (2024), p. 018. DOI: 10.21468/SciPostPhys.16.1.018. arXiv: 2303.05376 [hep-ph].
- [248] Vinicius Mikuni, Benjamin Nachman, and Mariel Pettee. “Fast point cloud generation with diffusion models in high energy physics”. In: *Phys. Rev. D* 108.3 (2023), p. 036025. DOI: 10.1103/PhysRevD.108.036025. arXiv: 2304.01266 [hep-ph].
- [249] Matthew Leigh, Debajyoti Sengupta, John Andrew Raine, Guillaume Quétant, and Tobias Golling. “Faster diffusion model with improved quality for particle cloud generation”. In: *Phys. Rev. D* 109.1 (2024), p. 012010. DOI: 10.1103/PhysRevD.109.012010. arXiv: 2307.06836 [hep-ex].
- [250] Claudius Krause and David Shih. “Fast and accurate simulations of calorimeter showers with normalizing flows”. In: *Phys. Rev. D* 107.11 (2023), p. 113003. DOI: 10.1103/PhysRevD.107.113003. arXiv: 2106.05285 [physics.ins-det].
- [251] Thorben Finke, Michael Krämer, Alexander Mück, and Jan Tönshoff. “Learning the language of QCD jets with transformers”. In: *JHEP* 06 (2023), p. 184. DOI: 10.1007/JHEP06(2023)184. arXiv: 2303.07364 [hep-ph].
- [252] Anja Butter, Nathan Huetsch, Sofia Palacios Schweitzer, Tilman Plehn, Peter Sorenson, and Jonas Spinner. “Jet diffusion versus JetGPT – Modern networks for the LHC”. In: *SciPost Phys. Core* 8 (2025), p. 026. DOI: 10.21468/SciPostPhysCore.8.1.026. arXiv: 2305.10475 [hep-ph].
- [253] CMS Collaboration. *JetHT primary dataset in MINIAOD format from RunG of 2016 (/JetHT/Run2016G-UL2016_MiniAODv2-v2/MINIAOD)*. 2024. DOI: 10.7483/OPE-NDATA.CMS.1KTG.X0W4. URL: <http://opendata.cern.ch/record/30508>.

-
- [254] CMS Collaboration. *JetHT primary dataset in MINIAOD format from RunH of 2016* (*/JetHT/Run2016H-UL2016_MiniAODv2-v2/MINIAOD*). 2024. DOI: 10.7483/OPENDATA.CMS.LT9E.T7RQ. URL: <http://opendata.cern.ch/record/30541>.
 - [255] Oz Amram, Luca Anzalone, Joschka Birk, Darius A. Faroughy, Anna Hallin, Gregor Kasieczka, Michael Krämer, Ian Pang, Humberto Reyes-Gonzalez, and David Shih. *Aspen Open Jets: a real-world ML-ready dataset for jet physics*. 2024. DOI: 10.25592/UHHFDM.16505. URL: <https://www.fdr.uni-hamburg.de/record/16505>.
 - [256] CMS Collaboration. *Jet primary dataset in AOD format from RunB of 2010* (*/Jet/Run2010B-Apr21ReReco-v1/AOD*). 2014. DOI: 10.7483/OPENDATA.CMS.3S7F.2E9W. URL: <http://opendata.cern.ch/record/5>.
 - [257] CMS Collaboration. *MultiJet primary dataset in AOD format from RunA of 2011* (*/MultiJet/Run2011A-12Oct2013-v1/AOD*). 2016. DOI: 10.7483/OPENDATA.CMS.8N95.GCTN. URL: <http://opendata.cern.ch/record/28>.
 - [258] CMS Collaboration. *MultiJet primary dataset in AOD format from Run of 2012* (*/MultiJet/Run2012A-22Jan2013-v1/AOD*). 2022. DOI: 10.7483/OPENDATA.CMS.WING.7QKV. URL: <http://opendata.cern.ch/record/24412>.
 - [259] ATLAS Collaboration. *DAOD_PHYSLITE format 2015-2016 Open Data for Research from the ATLAS experiment*. 2024. DOI: 10.7483/OPENDATA.ATLAS.9HK7.P5SI. URL: <http://opendata.cern.ch/record/80020>.
 - [260] Tomoe Kishimoto, Masahiro Morinaga, Masahiko Saito, and Junichi Tanaka. “Pre-training strategy using real particle collision data for event classification in collider physics”. In: *37th Conference on Neural Information Processing Systems*. Dec. 2023. arXiv: 2312.06909 [hep-ex].
 - [261] Giovanni Petrucciani, Andrea Rizzi, and Carl Vossalo. “Mini-AOD: A New Analysis Data Format for CMS”. In: *J. Phys. Conf. Ser.* 664.7 (2015), p. 7. DOI: 10.1088/1742-6596/664/7/072052. arXiv: 1702.04685 [physics.ins-det].
 - [262] CMS Collaboration. *PFNano producer tool for CMS 2016*. 2024. URL: <https://opendata.cern.ch/record/12504>.
 - [263] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. *Deep Learning Scaling is Predictable, Empirically*. 2017. arXiv: 1712.00409 [cs.LG].
 - [264] Oz Amram, Darius A. Faroughy, Tjarko Gerdes, Anna Hallin, Gregor Kasieczka, Michael Krämer, Humberto Reyes-Gonzalez, and David Shih. “Neural Scaling Laws for Jet Generation” (May 2026). arXiv: 2605.28940 [hep-ph].
 - [265] Joshua Batson and Yonatan Frederick Kahn. “Scaling laws in jet classification”. In: *SciPost Phys. Core* 8 (2025), p. 034. DOI: 10.21468/SciPostPhysCore.8.1.034. arXiv: 2312.02264 [hep-ph].
 - [266] Matthias Vigl, Nicole Hartman, Michael Kagan, and Lukas Heinrich. “Neural Scaling Laws for Boosted Jet Tagging”. In: *ICLR 2026 Workshop on Foundation Models for Science: Real-World Impact and Science-First Design*. 2026. arXiv: 2602.15781 [hep-ex].
 - [267] Jan-Lucas Uslu, Benjamin Nachman, and Christopher Ré. “Predict before you train: Scaling Laws for particle physics foundation models” (July 2026). arXiv: 2607.23377 [hep-ex].
-

REFERENCES

- [268] David Keetae Park et al. “FM4NPP: A Scaling Foundation Model for Nuclear and Particle Physics”. In: *The Fourteenth International Conference on Learning Representations*. 2026. arXiv: 2508.14087 [cs.LG].
- [269] Henning Bahl, Victor Bresó-Pla, Anja Butter, and Joaquín Iturriza Ramirez. “Scaling laws for amplitude surrogates” (Jan. 2026). arXiv: 2601.13308 [hep-ph].
- [270] Ang Lv, Kaiyi Zhang, Shufang Xie, Quan Tu, Yuhan Chen, Ji-Rong Wen, and Rui Yan. *An Analysis and Mitigation of the Reversal Curse*. 2024. arXiv: 2311.07468 [cs.CL].
- [271] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. “LLM2Vec: Large Language Models Are Secretly Powerful Text Encoders”. In: *First Conference on Language Modeling*. 2024. arXiv: 2404.05961 [cs.CL].
- [272] Lucas Georges Gabriel Charpentier and David Samuel. *GPT or BERT: why not both?* 2024. arXiv: 2410.24159 [cs.CL].
- [273] Xinru Yu, Bin Guo, Shiwei Luo, Jie Wang, Tao Ji, and Yuanbin Wu. *AntLM: Bridging Causal and Masked Language Models*. 2024. arXiv: 2412.03275 [cs.CL].
- [274] Hao Liu, Xinyang Geng, Lisa Lee, Igor Mordatch, Sergey Levine, Sharan Narang, and Pieter Abbeel. *Towards Better Few-Shot and Finetuning Performance with Forgetful Causal Language Models*. 2023. arXiv: 2210.13432 [cs.CL].
- [275] Hippolyte Gisserot-Boukhlef, Nicolas Boizard, Manuel Faysse, Duarte Miguel Alves, Emmanuel Malherbe, Andre Martins, CELINE HUDELOT, and Pierre Colombo. “Should We Still Pretrain Encoders with Masked Language Modeling?” In: *The Fourteenth International Conference on Learning Representations*. 2026. arXiv: 2507.00994 [cs.CL].
- [276] Adam Izdebski, Jan Olszewski, Pankhil Gawade, Krzysztof Koras, Serra Korkmaz, Valentin Rauscher, Jakub M. Tomczak, and Ewa Szczurek. “Synergistic Benefits of Joint Molecule Generation and Property Prediction”. In: *Transactions on Machine Learning Research* (2026). ISSN: 2835-8856. arXiv: 2504.16559 [cs.LG].
- [277] Ian Pang, Darius A. Faroughy, David Shih, Ranit Das, and Gregor Kasieczka. “SURF-ing to the Fundamental Limit of Jet Tagging” (Nov. 2025). arXiv: 2511.15779 [hep-ph].

List of Acronyms

SM	Standard Model
QED	Quantum Electrodynamics
QFT	Quantum Field Theory
QCD	Quantum Chromodynamics
VEV	vacuum expectation value
BSM	beyond the Standard Model
LHC	Large Hadron Collider
CERN	European Organization for Nuclear Research
IP	interaction point
HEP	high energy physics
ECAL	electromagnetic calorimeter
HCAL	hadronic calorimeter
MC	Monte Carlo
CKM	Cabibbo–Kobayashi–Maskawa
WIMP	weakly interacting massive particle
HiLumi-LHC	High-Luminosity Large Hadron Collider
PF	particle-flow
AI	artificial intelligence
ISR	initial state radiation
FSR	final state radiation
IRC	infrared and collinear
SLAC	SLAC National Accelerator Laboratory
BNL	Brookhaven National Laboratory
ML	machine learning
NN	neural network
MSE	mean squared error
MAE	mean absolute error
SGD	stochastic gradient descent

LIST OF ACRONYMS

MLP	multi-layer perceptron
ReLU	Rectified Linear Unit
EPiC	Equivariant Point Cloud
NLP	natural language processing
ODE	ordinary differential equation
CNF	continuous normalizing flow
OT	optimal-transport
CFM	conditional flow matching
NTP	next token prediction
LLM	large language model
VQ-VAE	vector-quantized variational autoencoder
FM	foundation model
PID	particle-ID
PV	primary vertex
MPM	Masked Particle Modeling
JEPA	Joint Embedding Predictive Architecture
DINO	Distillation with No Labels
EMA	exponential moving average
EPiC-FM	Equivariant Point Cloud Flow Matching
KDE	kernel density estimator
GAN	generative adversarial network
KL	Kullback-Leibler
AOJ	Aspen Open Jets
PET	point-edge transformer

Publications

Flow matching beyond kinematics: Generating jets with particle identification and trajectory displacement information

Joschka Birk^{1,*}, Erik Buhmann¹, Cedric Ewen¹, Gregor Kasieczka¹, and David Shih²

¹*Institute for Experimental Physics, Universität Hamburg,
Luruper Chaussee 149, 22761 Hamburg, Germany*

²*New High Energy Theory Center, Rutgers University, Piscataway, New Jersey 08854-8019, USA*



(Received 24 April 2024; accepted 27 February 2025; published 17 March 2025)

We introduce the first generative model trained on the JetClass dataset. Our model generates jets at the constituent level, and it is a permutation-equivariant continuous normalizing flow (CNF) trained with the flow matching technique. It is conditioned on the jet type, so that a single model can be used to generate the ten different jet types of JetClass. For the first time, we also introduce a generative model that goes beyond the kinematic features of jet constituents. The JetClass dataset includes more features, such as particle-ID and track impact parameter, and we demonstrate that our CNF can accurately model all of these additional features as well. Our generative model for JetClass expands on the versatility of existing jet generation techniques, enhancing their potential utility in high-energy physics research, and offering a more comprehensive understanding of the generated jets.

DOI: [10.1103/PhysRevD.111.052008](https://doi.org/10.1103/PhysRevD.111.052008)

I. INTRODUCTION

Recently there has been considerable interest and activity in generative modeling for jet constituents. While showering and hadronization with standard programs such as PYTHIA [1] and Herwig [2] are not a major computational bottleneck at the LHC [3], learning the properties of jets from data still has interesting potential applications. For example, generative modeling at the jet constituent level can be used to improve the performance of anomaly detection [4] techniques, since this allows to cover the full phase space instead of using solely high-level features that are calculated from the jet constituents.

More generally, learning jets is an interesting laboratory for method development. In particular, it has been fruitful and effective to view the jet constituents as a high-dimensional point cloud, and to devise methods for point cloud generative models that incorporate permutation invariance. This route has led to a number of state-of-the-art approaches, recently explored in [5–14], that combine different permutation-invariant layers such as transformers [15] and the equivariant point cloud (EPiC) layer [7], with state-of-the-art generative modeling frameworks such as diffusion [16–20] and flow matching [21–24]. Successful models developed for jet

point clouds can also potentially be adapted to other important point cloud generative modeling problems such as for fast emulation of Geant4 [25–27] calorimeter showers [11,13,28,29]. Finally, while event generation with generative models has concentrated primarily on low multiplicities and fixed structures [30–35], recent, permutation invariant approaches that allow for dynamic multiplicities exist as well [36,37].

So far, efforts for jet generation have focused almost exclusively on the JetNet dataset of Refs. [38,39]. Originally generated by [40], this dataset was subsequently adopted in the works of Ref. [5] as a benchmark dataset for jet generative modeling. However, the JetNet dataset has a number of drawbacks that are readily becoming apparent. First, its limited size (180 000 jets per jet type) means there are not enough jets in JetNet to facilitate the training of state-of-the-art generative models as well as metrics such as the binary classifier metric [41,42] which require additional training data. Second, JetNet uses small-radius ($R = 0.4$) jets (although the description in [5] incorrectly states a cone-size of $R = 0.8$ which is in disagreement with the observed angular distribution of constituents). This can lead to decay products not being fully contained in the jet, which can be seen, e.g., in distributions such as the jet mass for top quarks, where there is a prominent secondary mass peak (as can be seen, e.g., in [14], Fig. 4). Finally, JetNet focuses solely on the kinematics of the jet constituents, whereas there is a wealth of additional information inside the jets that could also be modeled, such as trajectory displacement, charge, and particle ID as illustrated in Fig. 1.

*Contact author: joschka.birk@uni-hamburg.de

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI. Funded by SCOAP³.

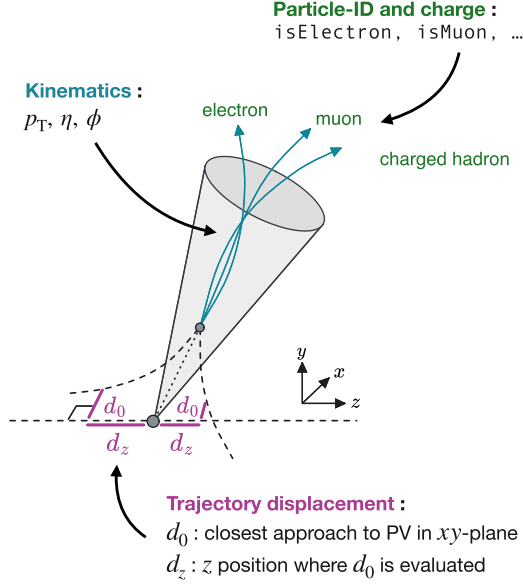


FIG. 1. Schematic overview of the different jet constituent features available in the JetClass dataset. The horizontal line at the bottom represents the beam axis and the circle on this line represents the primary vertex (PV).

In this work, we introduce the first generative model for jet constituents trained on the larger and more substantial JetClass dataset [43]. Other than demonstrating that existing techniques scale well to this new dataset, we also tackle new challenges introduced by the JetClass dataset, including the additional features mentioned above. We show that a single architecture consisting of a continuous normalizing flow (CNF) [44] constructed out of EPiC layers and trained with the flow matching technique [21–24] can learn to model all ten jet types and all features of JetClass well, both kinematics and beyond. Using the much larger statistics provided by JetClass we also train binary classifier metrics on the generative model for the first time, which allows us to further evaluate our results.

The rest of this paper is structured as follows: Sec. II describes the JetClass dataset in more detail; Sec. III introduces the used generative model; Sec. IV provides an overview of the generative performance; and finally Sec. V concludes this work.

II. DATASET

We use the JetClass dataset [43], originally introduced in Ref. [45]. This dataset encompasses various features both on jet level and on jet constituent level for ten different types of jets initiated by gluons and quarks (q/g), top quarks (t), as well as W , Z , and H bosons. Jets initiated by a top quark or a Higgs boson are further categorized based on their different decay channels, resulting in the following ten categories: q/g , $t \rightarrow bq\bar{q}'$, $t \rightarrow b\ell\nu$, $Z \rightarrow q\bar{q}$, $W \rightarrow qq'$, $H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, $H \rightarrow gg$, $H \rightarrow 4q$, and $H \rightarrow \ell\nu qq'$.

The jets are extracted from simulated events that are generated with MadGraph5_aMC@NLO [46]. Parton showering and hadronization is performed with PYTHIA [1] and detector effects are simulated with DELPHES [47] using the CMS detector [48] card. The jets are clustered with the anti- k_T algorithm [49] with a jet radius of $R = 0.8$. Jets are required to have a transverse momentum of $500 \text{ GeV} < p_T^{\text{jet}} < 1000 \text{ GeV}$ and a pseudorapidity of $|\eta^{\text{jet}}| < 2$. Furthermore, for all jets, except for the q/g jets, only those that contain all decay products of the boson or top quark are considered in the dataset.

Representative distributions of some of the features available in the JetClass dataset are presented in Fig. 2, illustrating substantial distinctions among the various jet categories. As can be seen in Fig. 2(a), the dataset covers jet types with a large variety of number of jet constituents. Additionally, Fig. 2(b) illustrates the expected tails of varying extents in the trajectory displacement. Notably, jet categories that do not include b quarks or c quarks have a rather narrow peak around 0, while jet categories that do include those types of quarks have a more spread out distribution, corresponding to the long lifetime of b - and c -hadrons.

The features that are used by our model are listed with the corresponding definitions in Table I. By conditioning our model on the jet type, we can train one model that is capable of generating all ten jet types. Furthermore, the jet transverse momentum p_T^{jet} and the jet pseudorapidity η^{jet} are used as conditioning features in order to be able to obtain the nonrelative constituent-level features after jet generation.

The kinematic features of the jet constituents are measured relative to the jet axis. i.e.,

$$\eta^{\text{rel}} = \eta^{\text{const}} - \eta^{\text{jet}} \quad (1)$$

$$\phi^{\text{rel}} = \phi^{\text{const}} - \phi^{\text{jet}} \quad (2)$$

$$p_T^{\text{rel}} = p_T^{\text{const}} / p_T^{\text{jet}}. \quad (3)$$

The trajectory displacement features as well as the discrete particle identification features are taken directly from the JetClass dataset. The jet constituents are assumed to be massless, as we do not include the particle energy in our model. In total, our model thus utilizes 13 constituent features to comprehensively characterize the constituents within generated jets. This set of features represents all available constituent features, from the JetClass dataset, with the only exception being the constituent energy.

III. MODEL

Overall, we follow the recent flow matching paradigm [21–24], using Gaussian probability paths as explained in Ref. [14].

The architecture of our model is based on the equivariant point cloud flow matching (EPiC-FM) model which was

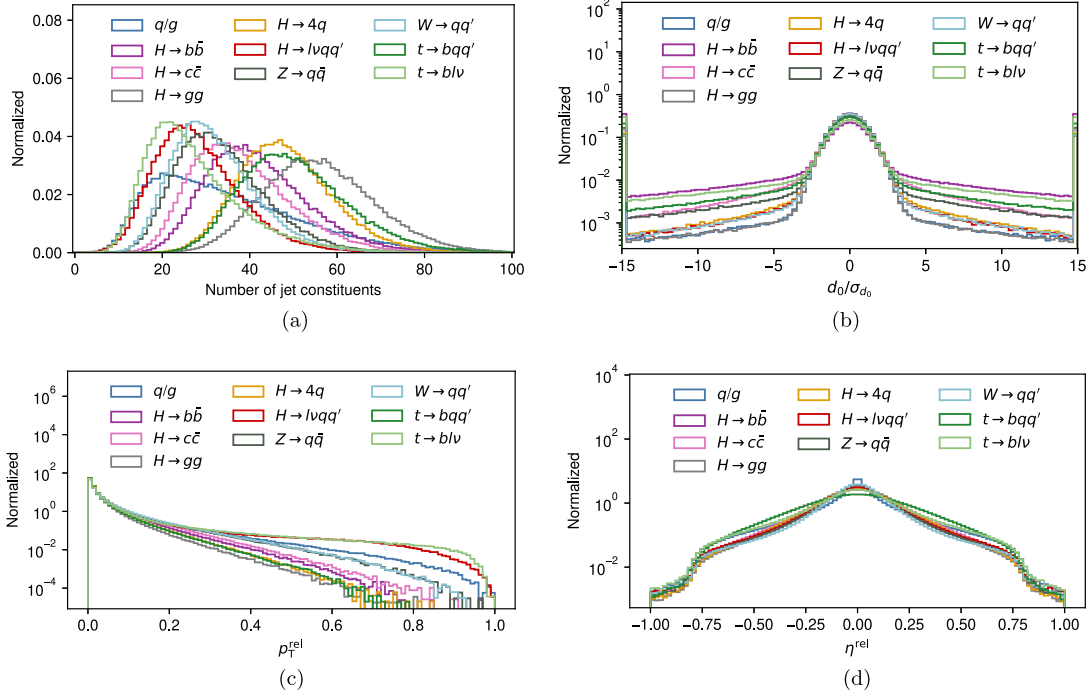


FIG. 2. Overview of some of the features from the JetClass dataset: (a) shows the number of jet constituents, (b) shows the significance of the transverse impact parameter d_0 , (c) shows the fraction of the jet p_T carried by the jet constituent, and (d) shows the difference η^{rel} between the constituent pseudorapidity and the jet axis. The impact parameter significance d_0/σ_{d_0} is only shown for charged particles since the impact parameter is 0 for neutral particles. The number of jet constituents is a jet-level feature our model is conditioned on while the remaining features correspond to constituent-level features that our model generates.

TABLE I. Features used for the studies in this paper. All features are either taken directly from the JetClass dataset or calculated from existing entries in the JetClass dataset. The model is conditioned on the jet features (first block) during generation. The remaining features are the constituent features our model is trained to generate.

Category	Variable	Definition
Jet features	Jet type	Indicating the type of the jet (i.e., $t \rightarrow bqq'$, q/g , ...)
	n_{const}	Number of constituents in the jet
	p_T^{jet}	Transverse momentum of the jet
	η^{jet}	Pseudorapidity of the jet
Kinematics	η^{rel}	Difference in pseudorapidity η between the constituent and the jet axis
	ϕ^{rel}	Difference in azimuthal angle ϕ between the constituent and the jet axis
	p_T^{rel}	Fraction of the constituent p_T and the jet p_T
Trajectory displacement	d_0	Transverse impact parameter value
	d_z	Longitudinal impact parameter value
	σ_{d_0}	Error of measured transverse impact parameter
	σ_{d_z}	Error of measured longitudinal impact parameter
Particle identification	Charge	Electric charge of the particle
	isChargedHadron	Flag if the particle is a charged hadron ($ \text{pid} ==211$ or 321 or 2212)
	isNeutralHadron	Flag if the particle is a neutral hadron ($ \text{pid} ==130$ or 2112 or 0)
	isPhoton	Flag if the particle is a photon ($\text{pid}==22$)
	isElectron	Flag if the particle is an electron ($ \text{pid} ==11$)
	isMuon	Flag if the particle is a muon ($ \text{pid} ==13$)

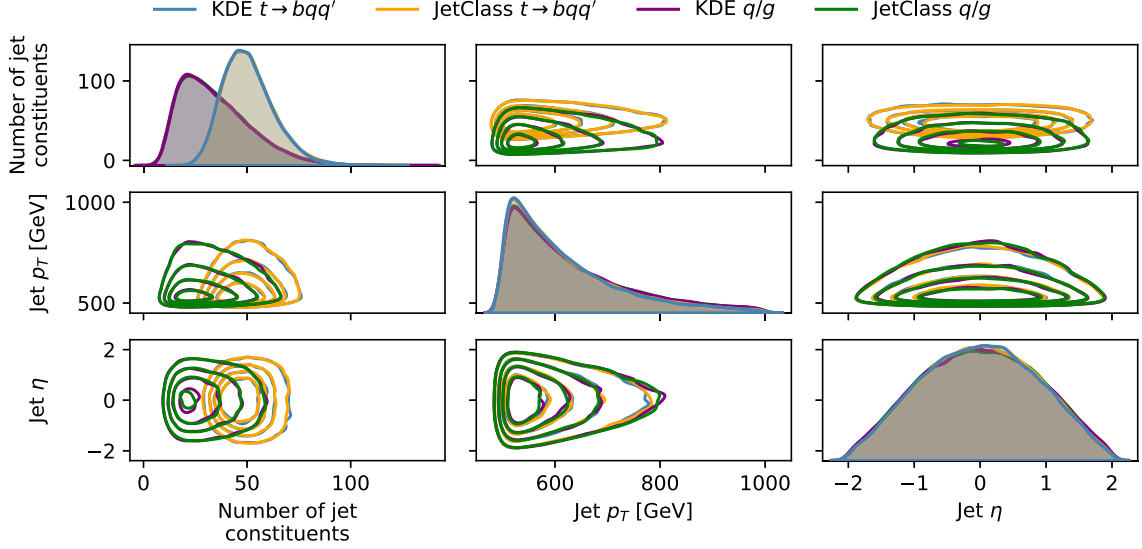


FIG. 3. Pair plot of the jet features of the JetClass dataset and the KDE samples for the $t \rightarrow bqq'$ and q/g jet types. The features generated by the KDE are used as conditioning features for the final generative model. The diagonal shows the distribution of the individual features, while the off-diagonal plots show the correlation between the features. The lines in the off-diagonal plots correspond to iso-proportions (20%, 40%, 60%, and 80%) of the density (e.g., 20% of the data points are outside the 20% line). Each of the 4 plotted datasets contains 100 000 jets.

introduced in Ref. [14]. It uses EPiC layers [7] to transform jet constituents as point clouds while preserving permutation symmetry. Besides the point cloud, EPiC layers also take as input a global context vector, which learns to encode higher-level information about the whole point cloud by using both mean and sum aggregation over the points. The use of the global context vector is responsible for the computational efficiency of the EPiC layers—unlike the quadratic scaling for architectures that include edge features between all pairs of points (such as transformers), the computational cost of EPiC layers only scales linearly with the number of points. Compared to the model in Ref. [14], the hyperparameters were adapted to obtain good performance with the increased complexity of the JetClass dataset: the number of EPiC layers was increased from 6 to 20, the embedding dimension was increased from 128 to 300, and the global vector dimension was increased from 10 to 16. A full list of the specific hyperparameters of the model is summarized in Appendix A, Table V.

The jet-type conditioning is realized with a one-hot encoding, while the jet transverse momentum p_T^{jet} and jet pseudorapidity η^{jet} are used without any preprocessing. For jet generation, we fit a kernel density estimator (KDE) to the jet features (p_T^{jet} , η^{jet} , and n_{const}) for each jet type individually, and sample from the KDE to obtain the conditioning features and the number of constituents for the generated jets. Since the KDE samples are continuous, we round the number of constituents to the nearest integer and clip all three features to the minimum and maximum values found in the training dataset. A verification that the

KDE samples are representative of the JetClass dataset is shown in Fig. 3. For each jet, our model acts on a tensor of shape (N, C) , with $N = 128$ being the maximum number of jet constituents and C being the number of constituent features. To generate jets with fewer constituents ($n_{\text{const}} < 128$), we apply a binary mask throughout training; this ensures that the inputs and outputs of the EPiC layers are valid and in particular guarantees that the mean and sum aggregation steps within the EPiC layers are correctly normalized. The particle features are shifted to mean 0 and scaled to the same standard deviation. Furthermore, only particles with $|\eta^{\text{rel}}| < 1$ are considered to remove a few particles in the tails of this distribution.¹

The model is trained for 500 epochs with a training sample size of 3M jets (300 000 jets per jet type).²

The AdamW optimizer [50] is used with a maximum learning rate of 0.001 and weight decay of 5×10^{-5} . The learning rate is scheduled with a cosine-annealing learning

¹The jets in the JetClass dataset are reported to correspond to a jet radius of $R = 0.8$, but some jet constituents have $|\eta^{\text{rel}}| > 0.8$. However, most of the constituents fall into $|\eta^{\text{rel}}| \leq 0.8$, so only a very small fraction of jet constituents is removed by this cut.

²It should be noted that this is just a small fraction (3%) of the training jets available in the JetClass dataset. Training with 5M instead of 3M jets did not show significant improvements in some preliminary trials. Given this, and the large potential computing expense from scaling up, we limited the training set size to this small fraction of JetClass in this first study. We still benefited from the larger size of JetClass in that it enabled us to train the binary classifier metrics.

rate scheduler with a linear warm-up period over 20 epochs. The model after the last epoch is chosen for evaluation.³ For the jet generation, we use the midpoint ODE solver, a 2nd order Runge-Kutta variant. After jet generation, the p_T^{rel} values of the generated jet constituents are clipped to the minimum and maximum values that are found in the training dataset. Furthermore, the electric charge is rounded to the nearest integer within $\{-1, 0, 1\}$. The remaining discrete particle-identification features are postprocessed with an argmax operation such that the particle-ID feature with the maximum value is set to 1 and the remaining ones are set to 0. This ensures that generated particles are unambiguously assigned to one particle type. It should be noted that this might not be the ideal handling of such discrete features, which could be improved and further investigated in future work. The model is implemented in PyTorch [51] using PyTorch lightning [52].

IV. RESULTS

In the following, we first evaluate the distributions of the constituent-level features that the model was trained on. Afterward, we evaluate jet-level features that are derived from the constituent-level features, namely the jet mass, the N -subjettiness [53] ratios⁴ τ_{32} and τ_{21} as well as the D_2 [55] observable, which is a ratio of energy correlation functions [56]. Lastly, we train a binary classifier as in Ref. [41] to distinguish between real jets from the JetClass dataset and jets generated by our model, which allows us to evaluate the quality of the generated jets further.

For the evaluation of our model, we generate 5M jets (i.e., 500 000 jets per jet type) and evaluate the different feature categories separately. Instead of using the Wasserstein-1 distance as a metric for the agreement between the generated and the target distribution as in [5,7,8], we are following [14] and using the Kullback-Leibler (KL) divergence $\text{KL}(p_{\text{JetClass}} || p_{\text{EPIC-FM}})$. As explained there, this metric choice is motivated by the fact that the Wasserstein-1 distance is well-suited for detecting distributions with different support, but can lead to misleading results for cases where the distributions have the same overlapping support but differ mainly in their shapes.

The KL divergence values are calculated by splitting the generated jets and the JetClass jets into 10 batches of 50 000 jets each and then calculate the KL divergence for each batch. This allows us to evaluate the uncertainty of the KL divergence to be able to quantitatively compare it to the

truth distribution. The reported values are the mean of those 10 individual KL divergence values and their standard deviation. For the truth values of the KL divergence, we take two independent samples of the test dataset and then calculate the KL divergence between those two samples using the same procedure as when comparing the real and generated jets. The bin edges of the binned probability distributions are constructed by calculating 100 equiprobable quantiles from the target distribution and setting the edges of the outermost bins to the minimum and maximum values found in the target distribution. Outliers of the distribution from the generated jets are put into the first and last bin.

A. Jet constituent modeling

A comparison of the kinematic features of the jet constituents is shown in Fig. 4. For this comparison, we chose to compare $t \rightarrow bqq'$ jets and q/g jets, since those two types of jets have distinct characteristics in both the η^{rel} and the p_T^{rel} distribution. While particles in $t \rightarrow bqq'$ jets have on average a larger η^{rel} and thus have an overall wider η^{rel} distribution, q/g jets are more collimated, resulting in a sharper η^{rel} peak around $\eta^{\text{rel}} = 0$. Concerning the p_T^{rel} distribution, $t \rightarrow bqq'$ jets are expected to show a smaller tail toward larger p_T^{rel} values, since $t \rightarrow bqq'$ jets contain on average more constituents and thus the jet p_T is distributed over more particles, leading to smaller p_T^{rel} values. The generated distributions of all three kinematic features agree very well with the corresponding distribution from the JetClass dataset, showing that our model is capable of generating jets of very different kinematic properties. This is also confirmed by the KL divergence values listed in Table II, which only have a small deviation from the truth values. In addition to that, Fig. 5 shows a pair plot of the kinematic features of the jet constituents, confirming that the model accurately captures the correlations between those features.

A comparison of the trajectory displacement modeling is shown in Fig. 4 for $H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, and $H \rightarrow gg$ jets. Due to the long lifetime of b - and c -hadrons, the trajectory displacement of jet constituents associated with those hadrons is expected to be on average larger than for other jet constituents. Thus, the trajectory displacement distributions of $H \rightarrow b\bar{b}$ and $H \rightarrow c\bar{c}$ jets are expected to be wider than the trajectory displacement distribution of $H \rightarrow gg$ jets. Since the trajectory displacement is by definition zero for neutral particles in the JetClass dataset, only charged particles are considered in the histograms in Fig. 4. For all three jet types, the histograms of the generated jets agree very well with the histograms of the real jets, showing that our model is capable of correctly modeling the trajectory displacement of the jet constituents. Notably, our model is able to catch the essential differences between the trajectory displacement distributions of

³Given that the training of this model is very stable, the last epoch essentially also corresponds to the epoch with the lowest loss. Thus, for simplicity the model from the last epoch was chosen.

⁴For the calculation of the subjettiness, the particles are clustered into two and three subjets with the k_t -algorithm and the winner-takes-all clustering scheme using SCIKIT-HEP/FastJet [54].

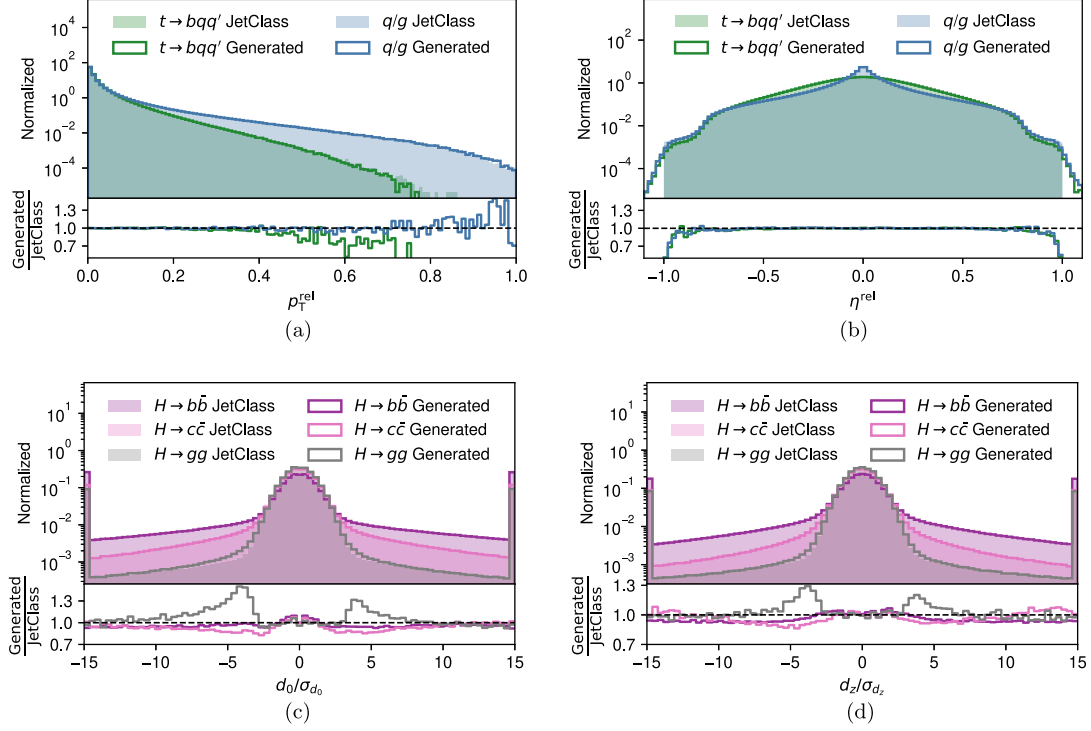


FIG. 4. Two kinematic features and the trajectory displacement of the jet constituents as generated by our model (solid lines) in comparison to the distributions obtained from the JetClass dataset (semi-transparent histograms). The upper row shows (a) the relative transverse momentum and (b) the relative pseudorapidity. The lower row shows (c) the transverse impact parameter significance and (d) the longitudinal impact parameter significance. Only charged particles are shown in the histograms of the trajectory displacement and the outermost bins show the underflow and overflow bins.

TABLE II. Kullback-Leibler divergence of constituent-level features with respect to the JetClass dataset.

	$\text{KL}^{p_T^{\text{rel}}} (\times 10^{-3})$	$\text{KL}^{\eta^{\text{rel}}} (\times 10^{-3})$	$\text{KL}^{d^{\text{rel}}} (\times 10^{-3})$	$\text{KL}^{d_0/\sigma_{d_0}} (\times 10^{-3})$	$\text{KL}^{d_z/\sigma_{d_z}} (\times 10^{-3})$
Truth (q/g)	0.05962 ± 0.01034	0.08033 ± 0.00970	0.07218 ± 0.01001	0.13298 ± 0.01427	0.11914 ± 0.01577
EPiC-FM (q/g)	0.07320 ± 0.00713	0.13035 ± 0.02168	0.11907 ± 0.01153	2.22425 ± 0.06336	3.90859 ± 0.08540
Truth ($H \rightarrow b\bar{b}$)	0.04807 ± 0.00391	0.07479 ± 0.01215	0.06021 ± 0.00821	0.10227 ± 0.01019	0.09546 ± 0.01107
EPiC-FM ($H \rightarrow b\bar{b}$)	0.07047 ± 0.00836	0.12195 ± 0.02243	0.11699 ± 0.02247	2.64120 ± 0.08700	1.48675 ± 0.07362
Truth ($H \rightarrow c\bar{c}$)	0.05415 ± 0.00703	0.07489 ± 0.01378	0.08457 ± 0.01507	0.10597 ± 0.01915	0.10944 ± 0.02522
EPiC-FM ($H \rightarrow c\bar{c}$)	0.07762 ± 0.00683	0.11306 ± 0.01941	0.13436 ± 0.00885	2.81678 ± 0.10350	2.03249 ± 0.08109
Truth ($H \rightarrow gg$)	0.03438 ± 0.00247	0.05780 ± 0.01550	0.05272 ± 0.00820	0.07463 ± 0.01375	0.07582 ± 0.00945
EPiC-FM ($H \rightarrow gg$)	0.05684 ± 0.00901	0.10059 ± 0.01345	0.10103 ± 0.01414	2.38777 ± 0.09853	3.75019 ± 0.13275
Truth ($H \rightarrow 4q$)	0.03950 ± 0.00466	0.05703 ± 0.00719	0.05324 ± 0.01082	0.08051 ± 0.01142	0.08549 ± 0.01939
EPiC-FM ($H \rightarrow 4q$)	0.05688 ± 0.00626	0.11775 ± 0.01993	0.10621 ± 0.01515	2.30474 ± 0.07151	3.37262 ± 0.12772
Truth ($H \rightarrow \ell\nu qq'$)	0.07116 ± 0.01293	0.09878 ± 0.02302	0.09697 ± 0.01441	0.13964 ± 0.01103	0.13760 ± 0.01663
EPiC-FM ($H \rightarrow \ell\nu qq'$)	0.10317 ± 0.01354	0.15427 ± 0.02272	0.13721 ± 0.00855	2.11118 ± 0.07543	2.70495 ± 0.09921
Truth ($Z \rightarrow q\bar{q}$)	0.06101 ± 0.00819	0.07686 ± 0.01193	0.07736 ± 0.01304	0.13846 ± 0.03211	0.13434 ± 0.01236
EPiC-FM ($Z \rightarrow q\bar{q}$)	0.07360 ± 0.00982	0.14488 ± 0.02458	0.12659 ± 0.01858	3.26740 ± 0.17081	2.17424 ± 0.13619
Truth ($W \rightarrow qq'$)	0.06281 ± 0.01084	0.08797 ± 0.00950	0.08648 ± 0.00847	0.12368 ± 0.01451	0.13684 ± 0.01732
EPiC-FM ($W \rightarrow qq'$)	0.08246 ± 0.00768	0.14039 ± 0.02249	0.13022 ± 0.01662	2.17685 ± 0.04841	2.70335 ± 0.11423
Truth ($t \rightarrow b\bar{q}q'$)	0.04260 ± 0.00667	0.05973 ± 0.00491	0.05715 ± 0.00749	0.07989 ± 0.01026	0.08392 ± 0.00982
EPiC-FM ($t \rightarrow b\bar{q}q'$)	0.08227 ± 0.01122	0.08847 ± 0.01358	0.09984 ± 0.01072	1.84127 ± 0.06797	1.86226 ± 0.10431
Truth ($t \rightarrow b\bar{\ell}\nu$)	0.07229 ± 0.01400	0.12652 ± 0.01816	0.14858 ± 0.03336	0.14958 ± 0.01513	0.15891 ± 0.02089
EPiC-FM ($t \rightarrow b\bar{\ell}\nu$)	0.10641 ± 0.01623	0.19671 ± 0.03365	0.18488 ± 0.05699	1.78050 ± 0.08955	1.48676 ± 0.06308

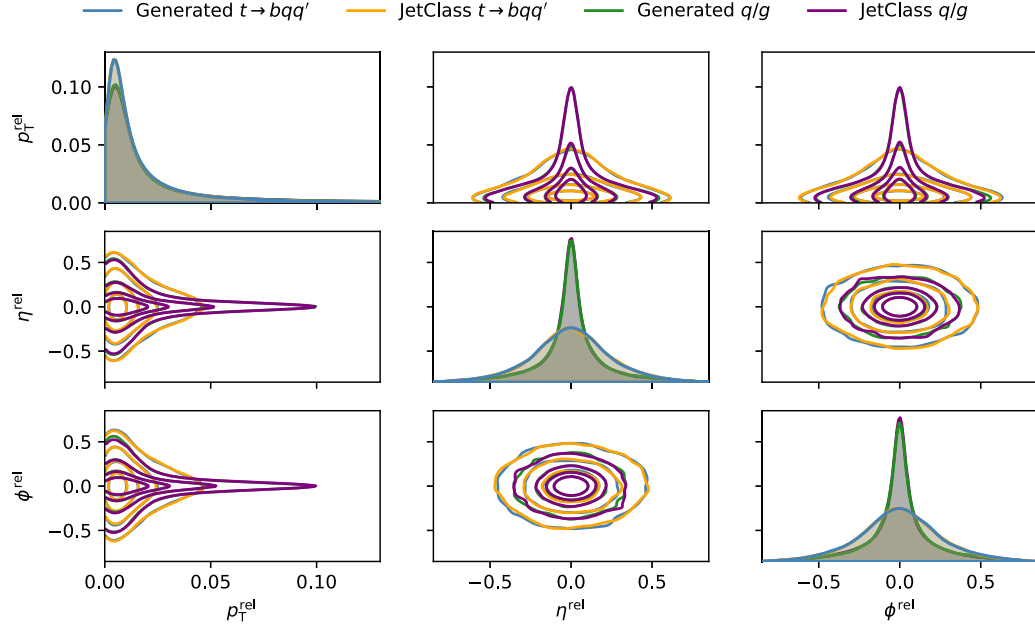


FIG. 5. Pair plot of the kinematic features of the jet constituents for the $t \rightarrow bqq'$ and q/g jet types illustrating that the model correctly captures the correlations between the different kinematic features. The lines in the off-diagonal plots correspond to isoproportions (20%, 40%, 60%, and 80%) of the density (e.g., 20% of the data points are outside the 20% line). Each of the 4 plotted datasets contains 100 000 constituents.

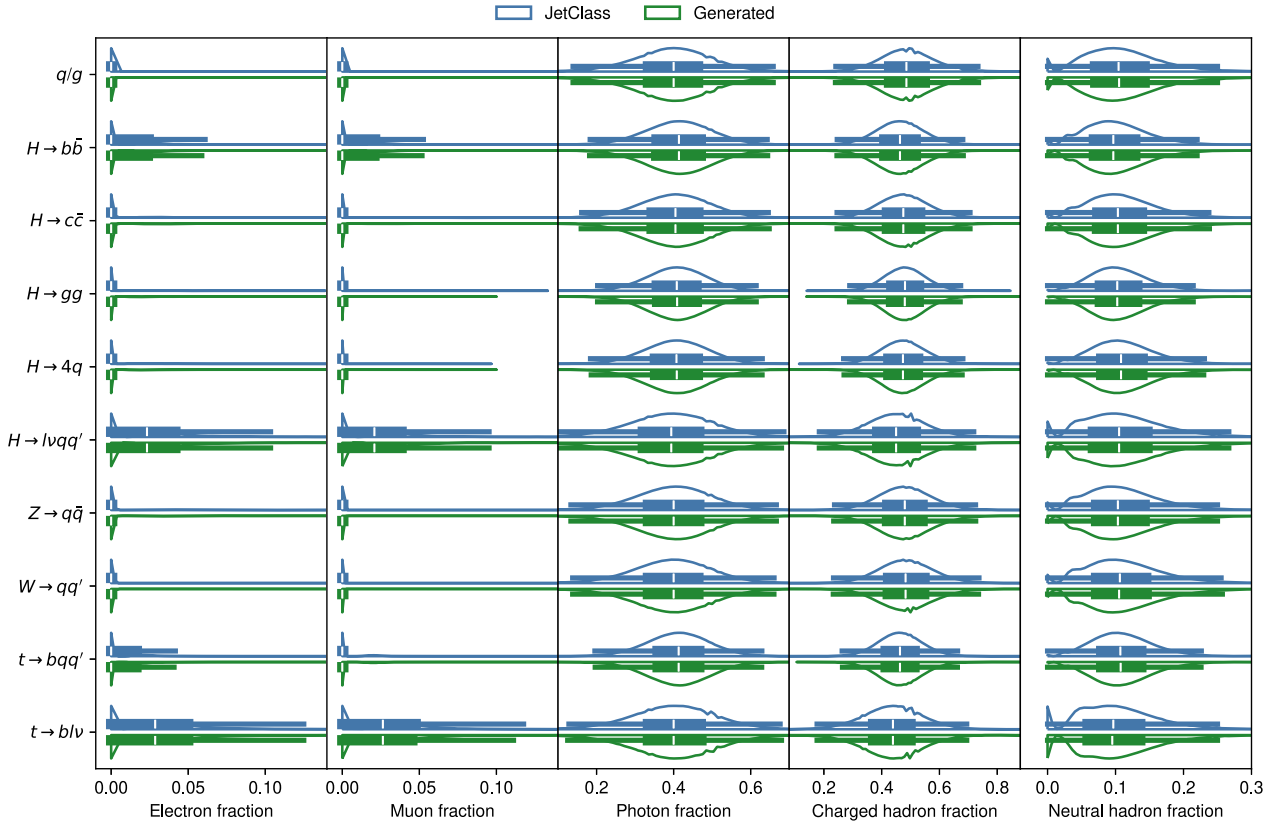


FIG. 6. The violin plots illustrate the distribution of fractions for various particle types within jet constituents. The box plots show the median, the quartiles, and 1.5 times the interquartile range, with a density plot to highlight distribution peaks. Notably, the electron and muon fractions predominantly peak at zero across most jet types, resulting in box plots positioned near zero.

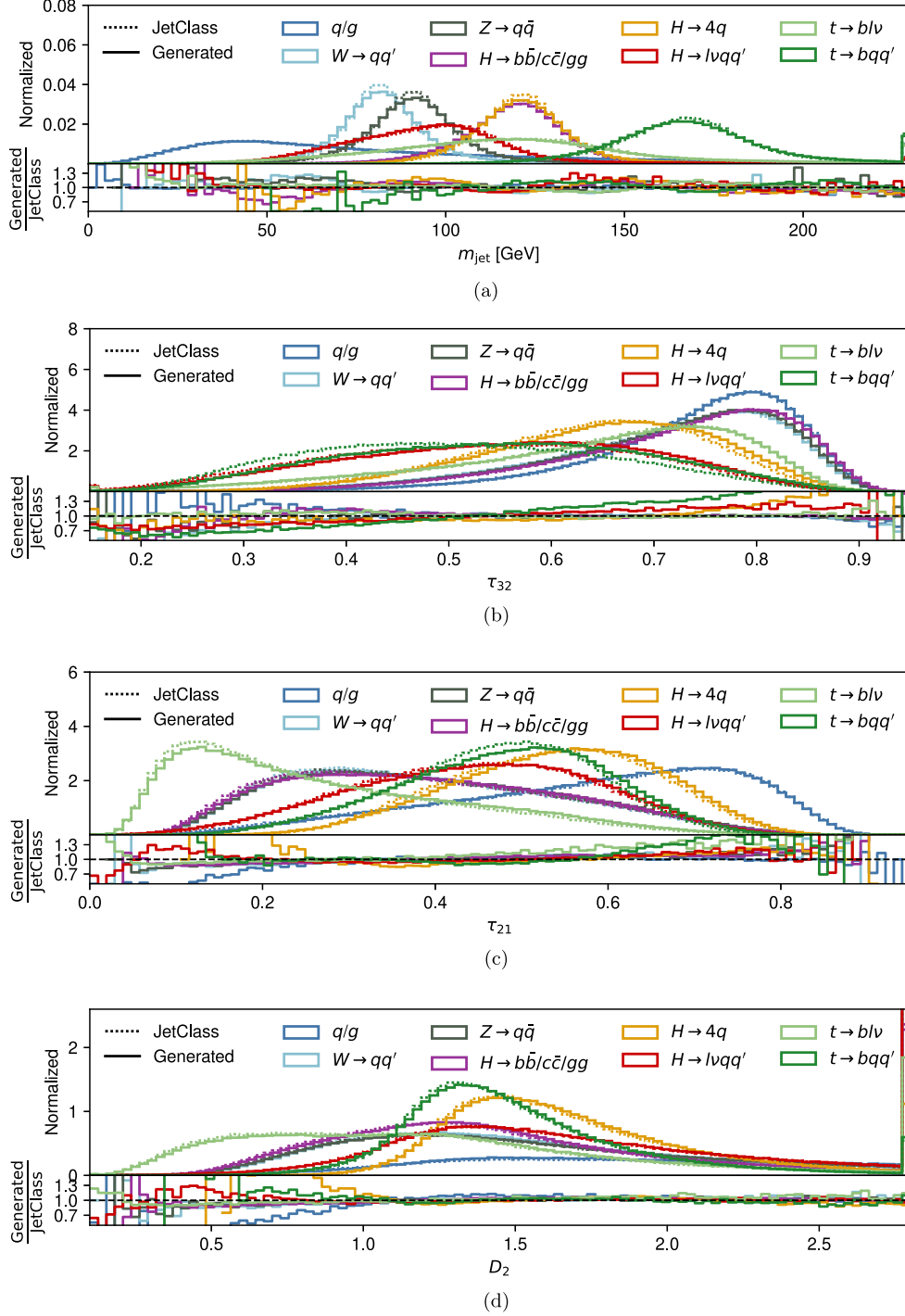


FIG. 7. Jet mass (a) and subjettness ratios τ_{32} (b), τ_{21} (c), and D_2 (d) for all ten jet types. The histograms in dotted lines show the distributions obtained from the JetClass dataset (i.e., real jets) and the solid lines show the histograms obtained from the generated jets. Jets from the categories $H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, and $H \rightarrow gg$ are grouped into one joint histogram for better readability, since the individual histograms show very similar shapes. All histograms include underflow and overflow bins.

$H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, and $H \rightarrow gg$ jets, which is an important feature from the physics point of view. However, as seen both in the ratio panels in Fig. 4 and in the corresponding KL divergence values in Table II, the agreement between

the target distribution and the distribution obtained from the generated jets is worse for the impact parameter features than for the kinematic features, showing that the modeling of these distributions is more challenging. This could be

TABLE III. Kullback-Leibler divergence of jet-level observables and Fréchet physics distance with respect to the JetClass dataset.

	$KL^{m_{\text{jet}}}(\times 10^{-3})$	$KL^{\tau_{21}}(\times 10^{-3})$	$KL^{\tau_{32}}(\times 10^{-3})$	$KL^{D_2}(\times 10^{-3})$	$FPD(\times 10^{-5})$
Truth (q/g)	1.96024 ± 0.22032	1.92403 ± 0.26225	2.01218 ± 0.27953	1.91381 ± 0.39724	7.33742 ± 3.30663
EPiC-FM (q/g)	2.02494 ± 0.32759	4.62940 ± 0.47603	2.27362 ± 0.25510	4.99645 ± 0.47051	7.91040 ± 1.94646
Truth ($H \rightarrow b\bar{b}$)	1.92982 ± 0.20477	1.94963 ± 0.21244	2.08080 ± 0.22548	1.97710 ± 0.22072	9.98274 ± 5.22338
EPiC-FM ($H \rightarrow b\bar{b}$)	3.24668 ± 0.46254	3.63796 ± 0.57295	2.68246 ± 0.34426	3.35243 ± 0.53867	12.07670 ± 6.57340
Truth ($H \rightarrow c\bar{c}$)	2.04281 ± 0.31760	2.00572 ± 0.22876	2.17238 ± 0.33699	1.90518 ± 0.17937	9.60921 ± 4.65058
EPiC-FM ($H \rightarrow c\bar{c}$)	4.75491 ± 0.45220	6.43288 ± 0.61917	3.05689 ± 0.48154	3.76940 ± 0.47839	14.32434 ± 7.91750
Truth ($H \rightarrow gg$)	1.96830 ± 0.19475	1.83727 ± 0.26071	1.88056 ± 0.23067	1.97745 ± 0.18155	8.65053 ± 6.80698
EPiC-FM ($H \rightarrow gg$)	3.74593 ± 0.40721	3.84047 ± 0.59242	2.80906 ± 0.43111	3.24857 ± 0.50123	13.46076 ± 6.72046
Truth ($H \rightarrow 4q$)	2.21949 ± 0.37219	2.09503 ± 0.30231	1.99994 ± 0.26632	1.86540 ± 0.31324	6.77410 ± 4.52764
EPiC-FM ($H \rightarrow 4q$)	5.17950 ± 0.48399	5.07132 ± 0.47784	7.51323 ± 0.66837	3.23005 ± 0.47515	6.72770 ± 1.95729
Truth ($H \rightarrow \ell\nu qq'$)	1.89667 ± 0.30666	1.93986 ± 0.26766	2.03454 ± 0.23668	1.98809 ± 0.20659	10.00879 ± 6.87615
EPiC-FM ($H \rightarrow \ell\nu qq'$)	3.48160 ± 0.48279	3.45981 ± 0.43930	9.25649 ± 0.88029	2.31713 ± 0.33259	13.79823 ± 4.53244
Truth ($Z \rightarrow q\bar{q}$)	2.06754 ± 0.17644	1.99159 ± 0.32238	1.95261 ± 0.23823	2.04952 ± 0.19436	7.52867 ± 3.61417
EPiC-FM ($Z \rightarrow q\bar{q}$)	4.81740 ± 0.54586	3.52108 ± 0.46385	1.96704 ± 0.21247	2.44368 ± 0.25113	10.14084 ± 5.14489
Truth ($W \rightarrow qq'$)	2.05795 ± 0.39753	2.04785 ± 0.22169	2.03512 ± 0.19298	1.99837 ± 0.24655	7.66854 ± 5.33720
EPiC-FM ($W \rightarrow qq'$)	5.74197 ± 0.43918	4.26782 ± 0.63239	2.67976 ± 0.41067	3.15276 ± 0.49860	12.76336 ± 7.56694
Truth ($t \rightarrow bqq'$)	1.86110 ± 0.25434	2.00490 ± 0.26157	2.06202 ± 0.24148	1.94072 ± 0.31298	7.45729 ± 3.61879
EPiC-FM ($t \rightarrow bqq'$)	4.67384 ± 0.60934	7.52936 ± 0.76181	30.02929 ± 1.07748	2.95281 ± 0.43531	9.22545 ± 3.82906
Truth ($t \rightarrow b\ell\nu$)	2.00893 ± 0.14388	1.85344 ± 0.22582	1.98487 ± 0.21507	2.00167 ± 0.27673	8.11577 ± 5.27167
EPiC-FM ($t \rightarrow b\ell\nu$)	3.38949 ± 0.23017	5.66341 ± 0.49231	2.56432 ± 0.36011	3.75036 ± 0.46000	7.96196 ± 4.10444

further optimized in future work by choosing a different preprocessing for the impact parameter features, by, e.g., transforming them using the hyperbolic tangent function, which would remove the large tails of the distributions.

The evaluation of the particle-ID modeling is shown in Fig. 6, where we show violin plots and box plots for the fraction of jet constituents of different particle types for all ten jet types. The agreement between the generated jets and the real jets is very good for all jet types, showing that on average the generated jets contain the same fraction of different particle types as the real jets. The modeling of the electric charge also shows very good agreement, which is shown in Appendix B.

B. Jet substructure modeling

The jet mass m_{jet} , the two subjettness ratios τ_{32} and τ_{21} , and D_2 are shown in Fig. 7 for the different jet types. The real jets are shown in dotted lines while the generated jets are shown in solid lines. Both the jet mass and the subjettness ratio distributions show very good agreement for all jet types. The largest deviations between the target distribution and the distribution of the generated jets are seen for $t \rightarrow bqq'$ and $H \rightarrow \ell\nu qq'$, where the distribution of the generated jets peaks at a larger value of τ_{32} . This mismodeling also shows in the values of the KL divergence which are listed in Table III for some of the jet-level observables. In addition to the 1-dimensional KL divergence values, we also evaluate the multidimensional distribution

by calculating the Fréchet physics distance (FPD) as introduced in [57], which is also listed in Table III. For the FPD calculation we follow the recommendations from [57], thus we evaluate the FPD based on a set of 36 EFPs (all EFPs of degree less than five) and use the JetNet [58] library to calculate both the EFPs and the corresponding FPD. Further studies were done to determine whether narrowing down our model's features to just kinematics and whether training solely on $t \rightarrow bqq'$ jets enhances the modeling of the $t \rightarrow bqq'$ substructure, which can be found in Appendix C.

C. Classifier test

In addition to the evaluation presented in the previous subsections, we also investigate the performance of our model with the classifier test proposed in Ref. [41]. Thus, a binary classifier is trained to distinguish between real jets from the JetClass dataset and generated jets that were generated with our model. This classifier test is performed for each jet type separately.

To train the classifier, we use the 500 000 generated jets and 500 000 real jets from the previous evaluation, and split them into 250 000, 50 000, and 200 000 jets for training, validation and testing of the classifier, respectively. The classifier is trained by fine-tuning the ParT-kin version from Ref. [45], which means that the last MLP of the ParT architecture is reinitialized and changed to two output nodes. The classifier is then trained with real vs generated

TABLE IV. AUC scores of the classifier test for the ten different jet types.

Jet type	AUC score
q/g	0.681
$W \rightarrow qq'$	0.683
$Z \rightarrow q\bar{q}$	0.689
$H \rightarrow \ell\nu qq'$	0.706
$t \rightarrow b\ell\nu$	0.706
$H \rightarrow b\bar{b}$	0.712
$H \rightarrow gg$	0.716
$H \rightarrow c\bar{c}$	0.728
$H \rightarrow 4q$	0.809
$t \rightarrow bqq'$	0.829

jets using the AdamW optimizer [50] with a learning rate of 0.001 for a few epochs. However, it should be noted that the classifier training was not extensively optimized. We noticed that the validation loss converges after just a few epochs after which the classifier starts to overfit. We therefore chose to use the model with the highest validation accuracy before clear signs of overfitting are visible.

The AUC values of the corresponding ROC curves are shown in Table IV. To estimate the uncertainty of the classifier scores, five classifiers are trained for $H \rightarrow b\bar{b}$ jets with different random seeds. All classifiers converge at similar performance, with the standard deviation of the AUC scores being $\sigma_{\text{AUC}}(H \rightarrow b\bar{b}) = 0.003$. The largest

AUC value is obtained for $t \rightarrow bqq'$ jets with an AUC of 0.829, confirming again that this is the most challenging jet type for our model. The results of the corresponding classifier test are shown in Fig. 8 in more detail. However, the AUC value for $t \rightarrow bqq'$ jets is still well below 1, allowing us to investigate the $t \rightarrow bqq'$ jets corresponding to different regions of the classifier output. Therefore, following Ref. [42], we split the $t \rightarrow bqq'$ jets into three regions, based on their weight $w = p_{\text{real}}/p_{\text{gen}}$ as illustrated in Figs. 8(a) and 8(b). The first region contains jets with $w > 2$, the second region contains jets with $0.5 < w < 2$, and the third region contains jets with $w < 0.5$. Thus, the outer regions correspond to jets that are clearly identified as either real or generated jets, while the center region contains jets where the classifier is not able to clearly distinguish them.

The corresponding subjettness ratios τ_{21} and τ_{32} are shown in Figs. 8(c) and 8(d) for $t \rightarrow bqq'$ jets from the outer regions. This shows the jet substructure differences between the $t \rightarrow bqq'$ jets that are clearly classified as either real or generated. $t \rightarrow bqq'$ jets that correspond to $w > 2$, i.e., jets of which the classifier is rather certain that they are real jets, have on average a smaller τ_{32} value than $t \rightarrow bqq'$ jets that are clearly identified as generated jets ($w < 0.5$). Therefore, the classifier classifies $t \rightarrow bqq'$ jets that are more 3-pronglike (i.e., with a smaller τ_{32} value) as real $t \rightarrow bqq'$ jets, while $t \rightarrow bqq'$ jets that are less 3-pronglike (i.e., with a larger τ_{32} value) tend to be

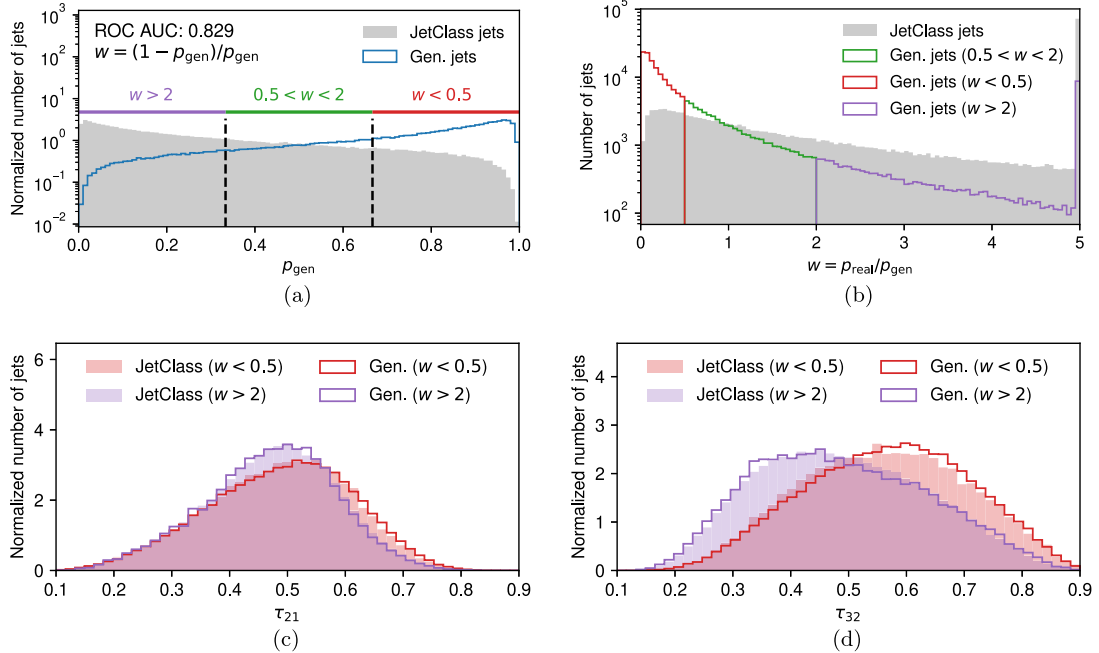


FIG. 8. Results of the classifier test for $t \rightarrow bqq'$ jets. (a) shows the classifier output for real and generated jets, as well as the cut values for the three regions. (b) shows the weight distribution of the generated jets with differently colored histograms for the generated jets from the different regions. (c) and (d) show the τ_{21} and τ_{32} distributions for the real jets and the generated jets from the regions where the classifier output distribution is dominated by real jets or generated jets.

classified as generated $t \rightarrow bqq'$ jets. Thus, our model seems to struggle with the modeling of the 3-prong structure of $t \rightarrow bqq'$ jets, which is also consistent with the results from the previous sections, where we saw that the τ_{32} distribution of the generated $t \rightarrow bqq'$ jets peaks at a slightly larger value than the τ_{32} distribution of the real $t \rightarrow bqq'$ jets. The peak at the larger τ_{32} value and thus the overall shift of the distribution is caused by the poorly modeled jets: given that 61% of the generated $t \rightarrow bqq'$ jets have $w < 0.5$, the overall τ_{32} distribution of the generated jets is dominated by jets with larger τ_{32} . On the other hand, only 15% of the real $t \rightarrow bqq'$ jets have $w < 0.5$, and 58% of the real $t \rightarrow bqq'$ jets have $w > 2$, which correspond to the purple distribution in Fig. 8(d).

V. CONCLUSION

Generative surrogate models of different physical processes are a powerful new tool with multiple potential applications. So far, speeding up classical simulations, either of particle interactions in calorimeters or of the calculation of matrix elements, has been the primary focus of research. However, learning surrogate models of jets has its own interesting aspects: (i) such models can, in principle, be trained on collider data, allowing the interpolation of distributions to estimate backgrounds; (ii) they can form parts of a fully differentiable analysis and interpretation chain; (iii) an interpretable version trained directly on collider data may provide an additional way of gaining insight into showering and hadronization processes; (iv) learning jets at the constituent level is a crucial stepping stone to simulating full events at the finest granularity; and (v) they provide a meaningful testbed to benchmark progress in generative models for point clouds.

The main result of this work is that the EPiC-FM [14] framework can be easily scaled up to more complex datasets. The move from the JetNet to the JetClass dataset results in more classes (10 instead of 5), more features per jet (13 instead of 3), geometrically larger jets ($R = 0.8$ instead of $R = 0.4$), and finally more jets available for training with JetClass containing a factor of 70 more jets per jet type than JetNet.

Overall we observe an accurate description of physical quantities. For some individual features and classes, the agreement between generated data and ground truth is on the same level as the statistical fluctuations between different samples drawn from the ground truth distribution. In general, more complex final states (i.e., multipronged signals with intermediate resonances) are more difficult to describe than relatively simple light quark or gluon-induced jets.

The capability to model quantities beyond jet kinematics demonstrated here will serve as an important factor in

further increasing the performance of anomaly detection algorithms. Previous results already showed that going from a few high-level to many low-level features increases the maximum achievable significance improvement by more than a factor of three [4]. Access to trajectory displacement and particle identification quantities now enable these models to be sensitive to final states including b -flavor and for example allow automatically covering long-lived [59] final states with anomaly detection as well, greatly increasing their accessible theory phase space.

Finally, the JetClass dataset has sufficient complexity to serve as a useful next benchmark for generative models applied to point-cloudlike data. Accordingly, we provide an extensive set of one-dimensional (KL-divergences) and multidimensional (classifier scores) evaluation methods for future reference and comparison.

ACKNOWLEDGMENTS

The authors thank Huilin Qu for his support with the JetClass dataset and helping with any questions. E. B. is funded by a scholarship of the Friedrich Naumann Foundation for Freedom and by the German Federal Ministry of Science and Research (BMBF) via Verbundprojekts 05H2018—R&D COMPUTING (Pilotmaßnahme ErUM-Data) Innovative Digitale Technologien für die Erforschung von Universum und Materie. J. B., E. B., C. E., and G. K. acknowledge support by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy—EXC 2121 Quantum Universe—390833306 and via the KISS consortium (05D23GU4) funded by the German Federal Ministry of Education and Research BMBF in the ErUM-Data action plan. D. S. is supported by DOE Grant No. DOE-SC0010008. This research was supported in part through the Maxwell computational resources operated at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany.

CODE AVAILABILITY

The code used to produce the results presented in this paper is available at [60].

APPENDIX A: HYPERPARAMETERS

The hyperparameters used for our generative model are listed in Table V. This choice of hyperparameters was found to lead to the best performance among several different configurations, though they were not extensively optimized.

The hyperparameters used for the classifier test are shown in Table VI. The model weights are initialized using the pre-trained ParT-kin model from [61].

TABLE V. Hyperparameters used for the baseline EPiC-FM model.

Hyperparameter	Value
Number of EPiC layers	20
Global dim. in EPiC layers	16
Hidden dim. in EPiC layers	300
ODE solver	midpoint
Number of function evaluations	200
Activation function	LeakyReLU(0.01)
AdamW [50] learning rate	0.001
Weight decay	0.00005
Learning rate scheduler	CosineAnnealing
Linear warm-up	20 epochs
Total number of epochs	EPiC-FM: 500 EPiC-FM-kin: 500 EPiC-FM-top: 1000 EPiC-FM-kintop: 1000
Batch size	1024
Trainable parameters	8 504 698
Training jets	3M (300 000 per jet type)

TABLE VI. Hyperparameters used for the classifier test with the ParT-kin [45] model.

Hyperparameter	Value
AdamW [50] learning rate	0.001
Weight decay	0.00005
Learning rate scheduler	Constant
Total number of epochs	100
Batch size	256
Trainable parameters	2 141 134
Training jets	500 000
Validation jets	100 000
Testing jets	400 000

APPENDIX B: MODELING OF THE ELECTRIC CHARGE

Similar to the evaluation of the particle-ID features, we show the distribution and the box plots for the fraction of particles corresponding to different electric charge values in Fig. 9.

While the spread is quite large as indicated by the intervals in Fig. 9, around half of the jet constituents are electrically neutral and correspondingly around a quarter of the constituents carries charge -1 and 1 , respectively.

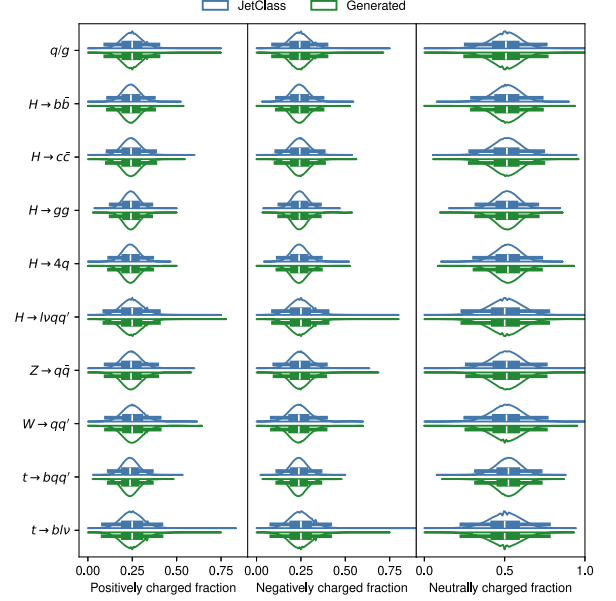


FIG. 9. The violin plots depict the distribution of the fraction of jet constituents with different electric charges. Each panel includes a box plot that highlights the median, quartiles, and 1.5 times the interquartile range.

APPENDIX C: FURTHER STUDIES FOR HADRONIC TOP JET MODELING

Since $t \rightarrow bqq'$ jets were found to be the most challenging jet type to model, further studies are performed to improve and investigate the modeling of those jets. To this end, we compare our baseline model, which we refer to in the following as EPiC-FM, to three additional models:

- (i) *EPiC-FM-top* is trained on the same features as the baseline model, but only the $t \rightarrow bqq'$ jets from the training dataset are used for training.
- (ii) *EPiC-FM-kin* is trained on kinematic features only (i.e., p_T^{rel} , η^{rel} , and ϕ^{rel}) but uses the full training dataset.
- (iii) *EPiC-FM-kintop* is trained on kinematic features only (i.e., p_T^{rel} , η^{rel} , and ϕ^{rel}) and only uses the $t \rightarrow bqq'$ jets from the training dataset.

The KL divergence values for the jet-level observables are listed for all models in Table VII and the substructure of the generated $t \rightarrow bqq'$ jets for the three different models is shown in Fig. 10. It can be seen that the models that are restricted to kinematic features show better agreement with the real jets than the two models that are trained on all

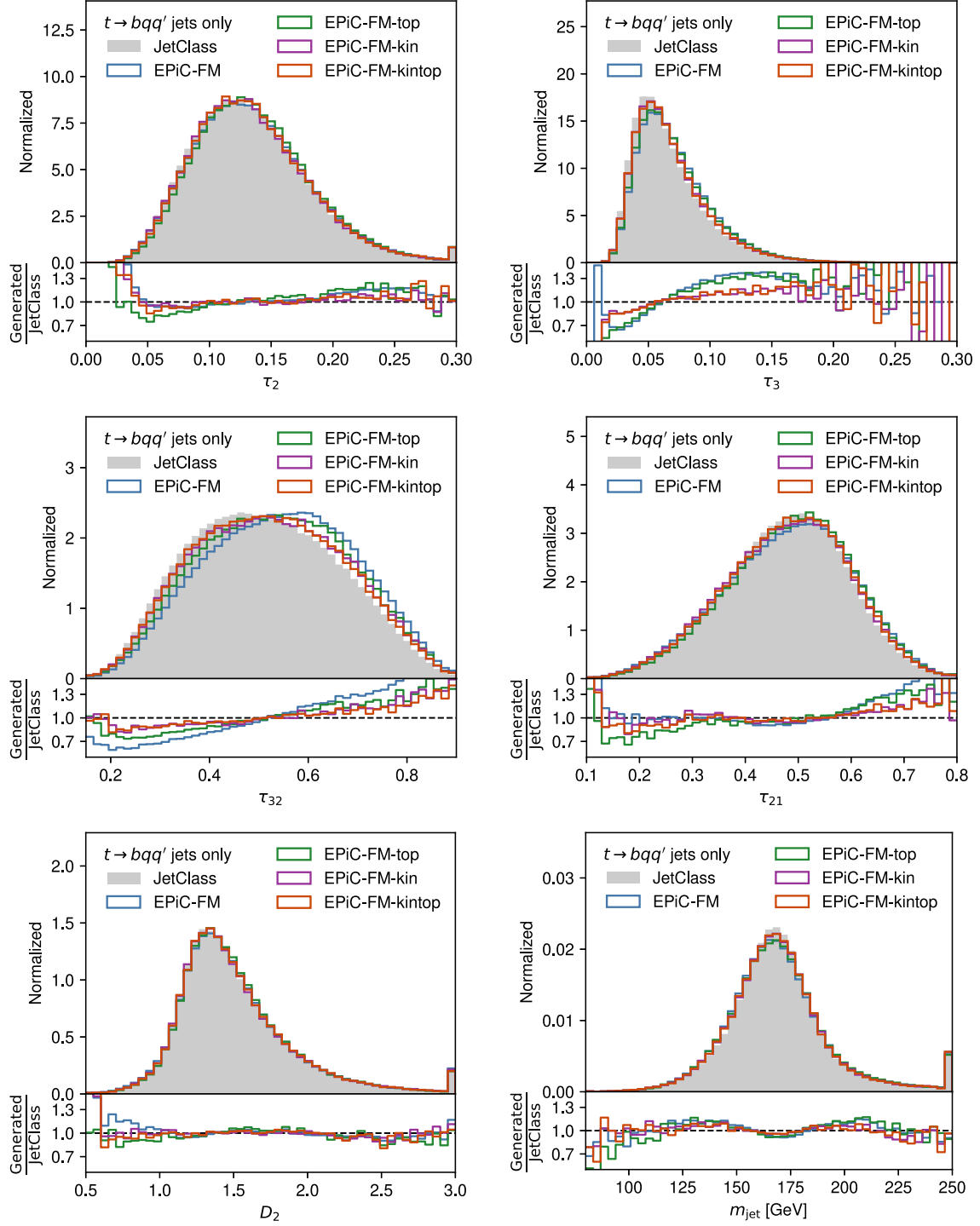


FIG. 10. Comparison of the jet substructure of $t \rightarrow bqq'$ jets for the four different models. The models EPiC-FM-top and EPiC-FM-kintop were trained on $t \rightarrow bqq'$ jets only.

features. This is also confirmed by the KL divergence, which show that the EPiC-FM-kin model has a significantly smaller KL divergence than the EPiC-FM model for all substructure observables. These observations lead to the conclusion that the additional features slightly confuse the

modeling of the kinematic features which results in a worse representation of the substructure. However, the jets are still modeled well in the case of all features, and the ability to model the discrete features extends the usability of such generative models.

TABLE VII. KL divergence metrics and FPD values for the four different models used for improving the modeling of $t \rightarrow bqq'$ jets. The models EPiC-FM-top and EPiC-FM-kintop were trained on $t \rightarrow bqq'$ jets only.

	$KL^{m_{\text{jet}}} (\times 10^{-3})$	$KL^{\tau_{21}} (\times 10^{-3})$	$KL^{\tau_{32}} (\times 10^{-3})$	$KL^{D_2} (\times 10^{-3})$	$FPD (\times 10^{-5})$
Truth ($t \rightarrow bqq'$)	1.86110 ± 0.25434	2.00490 ± 0.26157	2.06202 ± 0.24148	1.94072 ± 0.31298	7.45729 ± 3.61879
EPiC-FM ($t \rightarrow bqq'$)	4.67384 ± 0.60934	7.52936 ± 0.76181	30.02929 ± 1.07748	2.95281 ± 0.43531	9.22545 ± 3.82906
EPiC-FM-kin ($t \rightarrow bqq'$)	2.67246 ± 0.25860	3.13963 ± 0.37574	6.15219 ± 0.77443	2.10880 ± 0.33337	8.04998 ± 2.76330
EPiC-FM-top ($t \rightarrow bqq'$)	4.34749 ± 0.44568	7.60276 ± 0.58837	13.35373 ± 0.89500	3.16551 ± 0.25745	11.14224 ± 3.72043
EPiC-FM-kintop ($t \rightarrow bqq'$)	2.57447 ± 0.38647	3.48134 ± 0.31294	4.81889 ± 0.63065	2.33203 ± 0.34413	8.10681 ± 3.00268

The model that is restricted to $t \rightarrow bqq'$ jets and kinematic features, EPiC-FM-kintop, shows similar improvements as the EPiC-FM-kin model. However, while there is a slight improvement over the EPiC-FM-kin model in the KL divergence for τ_{32} , the KL divergence for τ_{21} and D_2 is slightly larger than for the kin-only model. In the case where all features are used, i.e., when comparing the EPiC-FM-top model to the baseline model EPiC-FM, we see an

improvement in τ_{32} , while not seeing an improvement in the KL divergence of τ_{21} and D_2 . Thus, the single jet-type model performs better than the model that is trained on all jet types if all features are used, while there is no clear improvement by training solely on $t \rightarrow bqq'$ jets if only kinematic features are used. All models show similar performance in the Fréchet physics distance (FPD) metric.

-
- [1] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands, An introduction to PYTHIA 8.2, *Comput. Phys. Commun.* **191**, 159 (2015).
 - [2] Manuel Bähr, Stefan Gieseke, Martyn A. Gigg, David Grellscheid, Keith Hamilton, Oluseyi Latunde-Dada, Simon Plätzer, Peter Richardson, Michael H. Seymour, Alexander Sherstnev, and Bryan R. Webber, Herwig++ physics and manual, *Eur. Phys. J. C* **58**, 639 (2008).
 - [3] Lyndon Evans and Philip Bryant, LHC machine, *J. Instrum.* **3**, S08001 (2008).
 - [4] Erik Buhmann, Cedric Ewen, Gregor Kasieczka, Vinicius Mikuni, Benjamin Nachman, and David Shih, Full phase space resonant anomaly detection, *Phys. Rev. D* **109**, 055015 (2024).
 - [5] Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos, Particle cloud generation with message passing generative adversarial networks, *arXiv:2106.11535*.
 - [6] Benno Käch, Dirk Krücker, Isabell Melzer-Pellmann, Moritz Scham, Simon Schnake, and Alexi Verney-Provatas, JetFlow: Generating jets with conditioned and mass constrained normalising flows, *arXiv:2211.13630*.
 - [7] Erik Buhmann, Gregor Kasieczka, and Jesse Thaler, EPiC-GAN: Equivariant point cloud generation for particle jets, *SciPost Phys.* **15**, 130 (2023).
 - [8] Matthew Leigh, Debajyoti Sengupta, Guillaume Quétant, John Andrew Raine, Knut Zoch, and Tobias Golling, PC-JeDi: Diffusion for particle cloud generation in high energy physics, *SciPost Phys.* **16**, 018 (2024).
 - [9] Vinicius Mikuni, Benjamin Nachman, and Mariel Pettee, Fast point cloud generation with diffusion models in high energy physics, *Phys. Rev. D* **108**, 036025 (2023).
 - [10] Benno Käch and Isabell Melzer-Pellmann, Attention to mean-fields for particle cloud generation, *arXiv:2305.15254*.
 - [11] Erik Buhmann, Sascha Diefenbacher, Engin Eren, Frank Gaede, Gregor Kasieczka, Anatolii Korol, William Korcari, Katja Krüger, and Peter McKeown, CaloClouds: Fast geometry-independent highly-granular calorimeter simulation, *J. Instrum.* **18**, P11025 (2023).
 - [12] Matthew Leigh, Debajyoti Sengupta, John Andrew Raine, Guillaume Quétant, and Tobias Golling, PC-Droid: Faster diffusion and improved quality for particle cloud generation, *Phys. Rev. D* **109**, 012010 (2024).
 - [13] Erik Buhmann, Frank Gaede, Gregor Kasieczka, Anatolii Korol, William Korcari, Katja Krüger, and Peter McKeown, CaloClouds II: Ultra-fast geometry-independent highly-granular calorimeter simulation, *J. Instrum.* **19**, P04020 (2024).
 - [14] Erik Buhmann, Cedric Ewen, Darius A. Faroughy, Tobias Golling, Gregor Kasieczka, Matthew Leigh, Guillaume Quétant, John Andrew Raine, Debajyoti Sengupta, and David Shih, EPiC-ly fast particle cloud generation with flow-matching and diffusion, *arXiv:2310.00049*.
 - [15] Ashish Vaswani *et al.*, Attention is all you need, in *Proceedings of Advances in Neural Information Processing Systems* (2017), Vol. 30, pp. 5999–6009, *arXiv:1706.03762*.
 - [16] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli, Deep unsupervised learning using non-equilibrium thermodynamics, *arXiv:1503.03585*.
 - [17] Yang Song and Stefano Ermon, Generative modeling by estimating gradients of the data distribution, *arXiv:1907.05600*.

- [18] Yang Song and Stefano Ermon, Improved techniques for training score-based generative models, [arXiv:2006.09011](#).
- [19] Jonathan Ho, Ajay Jain, and Pieter Abbeel, Denoising diffusion probabilistic models, [arXiv:2006.11239](#).
- [20] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole, Score-based generative modeling through stochastic differential equations, [arXiv:2011.13456](#).
- [21] Xingchao Liu, Chengyue Gong, and Qiang Liu, Flow straight and fast: Learning to generate and transfer data with rectified flow, [arXiv:2209.03003](#).
- [22] Michael S. Albergo and Eric Vanden-Eijnden, Building normalizing flows with stochastic interpolants, [arXiv:2209.15571](#).
- [23] Yaron Lipman, Ricky T.Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le, Flow matching for generative modeling, [arXiv:2210.02747](#).
- [24] Alexander Tong, Nikolay Malkin, Guillaume Huguët, Yanlei Zhang, Jarrod Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio, Improving and generalizing flow-based generative models with minibatch optimal transport, [arXiv:2302.00482](#).
- [25] S. Agostinelli *et al.*, Geant4—A simulation toolkit, *Nucl. Instrum. Methods Phys. Res., Sect. A* **506**, 250 (2003).
- [26] John Allison *et al.*, Geant4 developments and applications, *IEEE Trans. Nucl. Sci.* **53**, 270 (2006).
- [27] J. Allison *et al.*, Recent developments in Geant4, *Nucl. Instrum. Methods Phys. Res., Sect. A* **835**, 186 (2016).
- [28] Fernando Torales Acosta, Vinicius Mikuni, Benjamin Nachman, Miguel Arratia, Bishnu Karki, Ryan Milton, Piyush Karande, and Aaron Angerami, Comparison of point cloud and image-based models for calorimeter fast simulation, *J. Instrum.* **19**, P05003 (2024).
- [29] Moritz Alfons Wilhelm Scham, Dirk Krücker, Benno Käch, and Kerstin Borras, DeepTreeGAN: Fast generation of high dimensional point clouds, *EPJ Web Conf.* **295**, 09010 (2024).
- [30] Sydney Otten, Sascha Caron, Wieske de Swart, Melissa van Beekveld, Luc Hendriks, Caspar van Leeuwen, Damian Podareanu, Roberto Ruiz de Austri, and Rob Verheyen, Event generation and statistical sampling for physics with deep generative models and a density information buffer, *Nat. Commun.* **12**, 2985 (2021).
- [31] Bobak Hashemi, Nick Amin, Kaustuv Datta, Dominick Olivito, and Maurizio Pierini, LHC analysis-specific datasets with generative adversarial networks, [arXiv:1901.05282](#).
- [32] Riccardo Di Sipio, Michele Fucci Giannelli, Sana Ketabchi Haghighat, and Serena Palazzo, DijetGAN: A generative-adversarial network approach for the simulation of QCD dijet events at the LHC, *J. High Energy Phys.* **08** (2019) 110.
- [33] Anja Butter, Tilman Plehn, and Ramon Winterhalder, How to GAN LHC events, *SciPost Phys.* **7**, 075 (2019).
- [34] Yasir Alanazi, Nobuo Sato, Tianbo Liu, Wally Melnitchouk, Pawel Ambrozewicz, Florian Hauenstein, Michelle P. Kuchera, Evan Pritchard, Michael Robertson, Ryan Strauss, Luisa Velasco, and Yaohang Li, Simulation of electron-proton scattering events by a feature-augmented and transformed generative adversarial network (FAT-GAN), in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, edited by Zhi-Hua Zhou (IJCAI, 2021), p. 2126, [arXiv:2001.11103](#).
- [35] Anja Butter, Theo Heimel, Sander Hummerich, Tobias Krebs, Tilman Plehn, Armand Rousselot, and Sophia Vent, Generative networks for precision enthusiasts, *SciPost Phys.* **14**, 078 (2023).
- [36] Anja Butter, Nathan Huetsch, Sofia Palacios Schweitzer, Tilman Plehn, Peter Sorrenson, and Jonas Spinner, Jet Diffusion versus JetGPT—Modern Networks for the LHC, [arXiv:2305.10475](#).
- [37] Anja Butter, Tomas Jezo, Michael Klasen, Mathias Kuschick, Sofia Palacios Schweitzer, and Tilman Plehn, Kicking it off(-shell) with direct diffusion, *SciPost Phys. Core* **7**, 064 (2024).
- [38] Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos, JetNet (2022).
- [39] Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos, JetNet150 (2022).
- [40] Evan Coleman, Marat Freytsis, Andreas Hinzmann, Meenakshi Narain, Jesse Thaler, Nhan Tran, and Caterina Vernieri, The importance of calorimetry for highly-boosted jet substructure, *J. Instrum.* **13**, T01003 (2018).
- [41] Claudius Krause and David Shih, CaloFlow: Fast and accurate simulations of calorimeter showers with normalizing flows, *Phys. Rev. D* **107**, 113003 (2023).
- [42] Ranit Das, Luigi Favaro, Theo Heimel, Claudius Krause, Tilman Plehn, and David Shih, How to understand limitations of generative networks, *SciPost Phys.* **16**, 031 (2024).
- [43] Huilin Qu, Congqiao Li, and Sitian Qian, JetClass: A large-scale dataset for deep learning in jet physics (2022).
- [44] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud, Neural ordinary differential equations, [arXiv:1806.07366](#).
- [45] Huilin Qu, Congqiao Li, and Sitian Qian, Particle transformer for jet tagging, in *Proceedings of the 39th International Conference on Machine Learning* (2022), pp. 18281–18292, [arXiv:2202.03772](#).
- [46] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations, *J. High Energy Phys.* **07** (2014) 079.
- [47] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi, DELPHES 3: A modular framework for fast simulation of a generic collider experiment, *J. High Energy Phys.* **02** (2014) 057.
- [48] The CMS Collaboration, The CMS experiment at the CERN LHC, *J. Instrum.* **3**, S08004 (2008).
- [49] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez, The anti- k_t jet clustering algorithm, *J. High Energy Phys.* **04** (2008) 063.
- [50] Ilya Loshchilov and Frank Hutter, Decoupled weight decay regularization, [arXiv:1711.05101](#).
- [51] Adam Paszke *et al.*, PyTorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., Vancouver, CA, 2019), pp. 8024–8035.

- [52] William Falcon and The PyTorch Lightning Team, PyTorch lightning (2019).
- [53] Jesse Thaler and Ken Van Tilburg, Identifying boosted objects with n-subjettiness, *J. High Energy Phys.* **03** (2011) 015.
- [54] Aryan Roy, Jim Pivarski, Chris Papageorgakis, Javier Duarte, Lindsey Gray, Henry Schreiner, Raghav Kansal, Matthew Feickert, Kilian Lieret, and Srothman, SCIKIT-HEP/FastJet: V3.4.0.4 (2023), <https://zenodo.org/records/7254447>.
- [55] Andrew J. Larkoski, Ian Mout, and Duff Neill, Analytic boosted boson discrimination, *J. High Energy Phys.* **05** (2016) 117.
- [56] Andrew J. Larkoski, Gavin P. Salam, and Jesse Thaler, Energy correlation functions for jet substructure, *J. High Energy Phys.* **06** (2013) 108.
- [57] Raghav Kansal, Anni Li, Javier Duarte, Nadezda Chernyavskaya, Maurizio Pierini, Breno Orzari, and Thiago Tomei, Evaluating generative models in high energy physics, *Phys. Rev. D* **107**, 076017 (2023).
- [58] Raghav Kansal, Carlos Pareja, Zichun Hao, and Javier Duarte, JetNet: A Python package for accessing open datasets and benchmarking machine learning methods in high energy physics, *J. Open Source Software* **8**, 5789 (2023).
- [59] Juliette Alimena *et al.*, Searching for long-lived particles beyond the standard model at the Large Hadron Collider, *J. Phys. G* **47**, 090501 (2020).
- [60] https://github.com/uhh-pd-ml/beyond_kinematics
- [61] https://github.com/jet-universe/particle_transformer



PAPER

OPEN ACCESS

RECEIVED
11 April 2024REVISED
4 June 2024ACCEPTED FOR PUBLICATION
23 July 2024PUBLISHED
2 August 2024

Original Content from
this work may be used
under the terms of the
[Creative Commons
Attribution 4.0 licence](#).

Any further distribution
of this work must
maintain attribution to
the author(s) and the title
of the work, journal
citation and DOI.

OMNIJET- α : the first cross-task foundation model for particle physicsJoschka Birk^{*} , Anna Hallin^{*} and Gregor Kasieczka

Institute for Experimental Physics, Universität Hamburg, Luruper Chaussee 149, 22761 Hamburg, Germany

^{*} Authors to whom any correspondence should be addressed.E-mail: joschka.birk@uni-hamburg.de and anna.hallin@uni-hamburg.de**Keywords:** foundation models, particle physics, transfer learning, generation, classification, multitask, transformer

Abstract

Foundation models are multi-dataset and multi-task machine learning methods that once pre-trained can be fine-tuned for a large variety of downstream applications. The successful development of such general-purpose models for physics data would be a major breakthrough as they could improve the achievable physics performance while at the same time drastically reduce the required amount of training time and data. We report significant progress on this challenge on several fronts. First, a comprehensive set of evaluation methods is introduced to judge the quality of an encoding from physics data into a representation suitable for the autoregressive generation of particle jets with transformer architectures (the common backbone of foundation models). These measures motivate the choice of a higher-fidelity tokenization compared to previous works. Finally, we demonstrate transfer learning between an unsupervised problem (jet generation) and a classic supervised task (jet tagging) with our new OMNIJET- α model. This is the first successful transfer between two different and actively studied classes of tasks and constitutes a major step in the building of foundation models for particle physics.

1. Introduction

Foundation models are a new class of machine learning models which are trained on broad datasets and problems and are able to generalize to a variety of downstream tasks and datasets [1]. Large-language models (LLMs) such as BERT [2], BART [3], GPT-3 [4], and LLaMA [5] are examples of foundation models for text data while DALL-E 2 [6] is an example of an image-based model.

The benefits of a foundation model for particle physics data would be significant: while machine learning models developed so far typically outperform classical approaches (and often by a large margin) [7, 8], available statistics for training these models is a constant issue [9, 10]. It becomes even more extreme in the case of searches for rare processes, where only a small fraction of simulated events might pass pre-selection criteria. Foundation models on the other hand are pre-trained and need fewer examples to be fine-tuned for a specific task [11].

Beyond improving the physics performance by e.g. increasing selection accuracy of classification tasks, foundation models also address the other major issue currently facing particle physics: limited computing resources in the face of an ever increasing amount of data [12, 13]. This problem has already spawned the development of e.g. increasingly high fidelity models for the simulation of calorimeters, as well as techniques for the speeding up of other Monte Carlo simulations [14–21]. By allowing the re-use of models across datasets and tasks, foundation models will play an important role in reducing this computational burden. Note that this effect will be further compounded by the potential for optimization in computing operations from one model used across multiple tasks.

Finally, using closely related architectures across different tasks inside one experiment, across experimental collaborations, and in the exchange with the theory community will make results easier to re-interpret [22, 23].

These potential benefits have inspired research into proto-foundation models suitable for particle physics. For example, [24–28] investigated how known physical symmetries could be used to learn powerful embeddings of jet data, [29] showed the versatility of graph-based message passing networks for datasets from different domains of physics, [30, 31] demonstrated conditioning generative models on the geometry of the detector to allow the simultaneous simulation of multiple detectors with one architecture, [32, 33] used meta-learning for mass-decorrelation and weak-supervision, and [10] achieved state-of-the-art performance on the top tagging landscape dataset [34] by pre-training on a different dataset [35] and transferring the results.

Due to their flexibility demonstrated across language and other domains, transformers [36] are currently the most suitable candidate architecture for building foundation models for applications in particle physics. Taking inspiration from LLMs where sentences are generated autoregressively, recent efforts have demonstrated success with autoregressive generation of particle physics data, for example using a transformer to generate $t \rightarrow bq\bar{q}'$ and q/g jets [37] or to generate Z +jets events [38]. In [39] it was demonstrated how a transformer backbone can be pre-trained using a BERT-like scheme where the model is trained to predict masked out jet constituents, resulting in an improvement of the performance when fine-tuning the backbone (with a new classification head) for jet tagging, especially at small training dataset sizes. Furthermore, [40] showed how a tokenized detector representation can be used in a BERT-like model for track reconstruction, and [41] used tokenization of detector images together with a transformer to capture the context within a collider event for subsequent generation¹.

In this work, we will explore whether an autoregressive generative pretrained transformer (GPT) model can be used as a foundation model for jet physics. However, the standard GPT constructions are not built to deal with continuous input data, but rather tokenized data. As point clouds are the most versatile representation of physics data [7, 16, 43–45] and can incorporate both event level information, jet substructure, and even low-level detector signals, finding a suitable input transformation for point clouds to tokens is the most pressing problem. Various tokenization strategies have been explored, for example using a simple mapping based on binning the input space in [37], a Gaussian mixture model in [38], and using an additional conditional embedding network in [39].

Here, we follow the conditional tokenization strategy from [39, 46, 47], but first take a step back to verify the quality and trade-offs involved in building these tokens. This will allow us to formulate quality measures to choose a suitable tokenization model, leading to an increase in codebook size from 512 tokens in [39] to 8192 tokens.

Using this representation, we will first demonstrate training a generative model for jets as tokens in an unsupervised way for the JETCLASS [35] dataset.

Finally, this allows us to test whether the information encoded in a model that was trained to generate jets can also be transferred to the task of classifying them. Observing such a transfer ability across different classes of tasks—as opposed to transfer between different classification or generation problems—would be a crucial ingredient to building foundation models for physics data, and has not yet been achieved. A graphical representation of this approach is provided in figure 1. As this is the first prototype of a model to tackle all tasks with jets in particle physics, it is named OMNIJET- α .

The rest of the paper is organized as follows: section 2 introduces the data as well as the tokenization approach, the generative architecture, and the transfer learning strategy. Next, section 3 shows the results of the tokenization study, the generative performance, as well as tests of the transfer learning capabilities of the model. Finally, section 4 summarizes the results and provides a brief outlook.

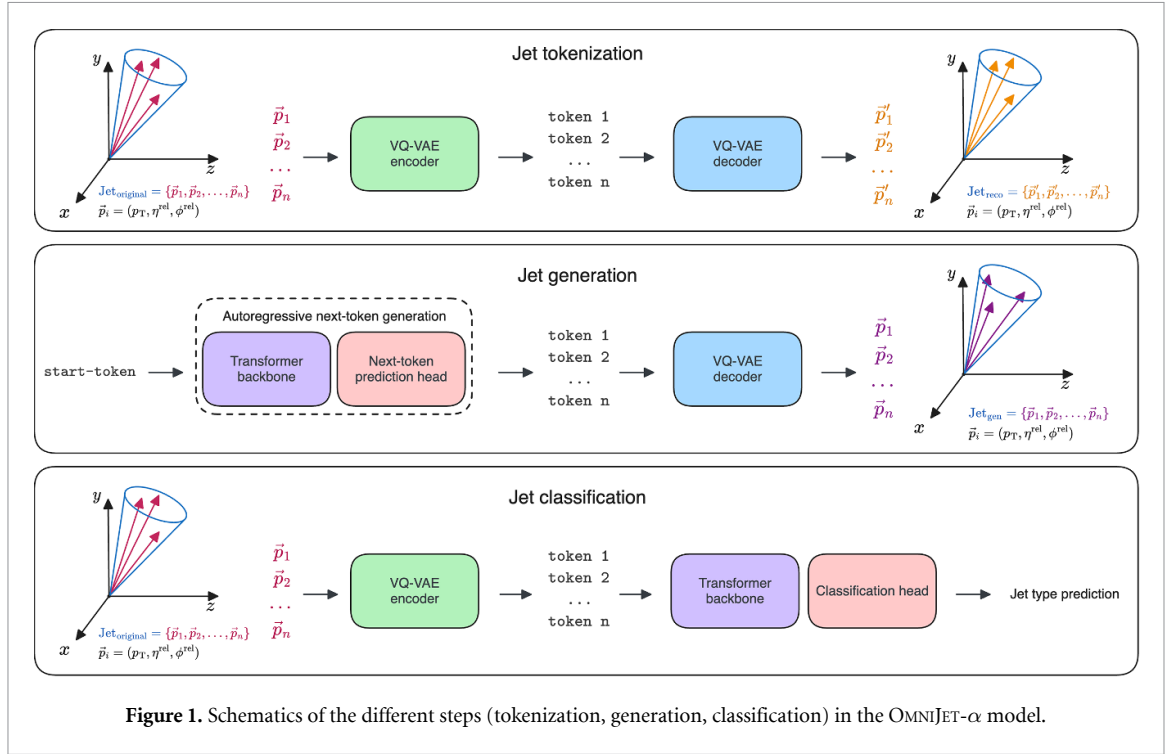
2. Methods and dataset

2.1. Dataset

All studies are performed using the JETCLASS dataset [35], originally introduced in [10]. It contains both jet-level and constituent-level features for ten different types of jets initiated by gluons and quarks (q/g), top quarks (t , subdivided by their decay mode into $t \rightarrow bq\bar{q}'$ and $t \rightarrow b\ell\nu$), as well as W , Z , and H ($H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, $H \rightarrow gg$, $H \rightarrow 4q$, and $H \rightarrow \ell\nu q\bar{q}'$) bosons.

Events are simulated using MADGRAPH5_AMC@NLO [48] with parton showering and hadronization done by PYTHIA [49]. A simplified detector simulation implemented in DELPHES [50] using the CMS detector [51] card is performed. Constituents are clustered into jets using the anti- k_T algorithm [52] with a distance parameter of $R = 0.8$.

¹ In addition, shortly after this paper was out, [42] demonstrated contrastive learning with re-simulated events as a suitable pre-training task.



Jets are selected if they have a transverse momentum of $500 \text{ GeV} < p_T^{\text{jet}} < 1000 \text{ GeV}$ and a pseudorapidity of $|\eta^{\text{jet}}| < 2$. Additionally, truth-level matching is performed for all classes except q/g and only jets that contain all the decay products of the boson or top quark are included. The resulting dataset contains 100M jets for training, 5M jets for validation, and 20M jets for testing, with equal numbers of jets for each class. This split into training, validation, and test sets corresponds to the default split used in the JETCLASS dataset, in order to allow for a straightforward comparison with previous and future works.

In this work, only the kinematic information per particle (p_T , ϕ , η) is used while the particle mass m is approximated as zero. Next, the azimuth angle ϕ and the pseudorapidity η are pre-processed to be relative to the jet axis²:

$$\eta^{\text{rel}} = \eta^{\text{particle}} - \eta^{\text{jet}} \quad (1)$$

$$\phi^{\text{rel}} = \phi^{\text{particle}} - \phi^{\text{jet}}. \quad (2)$$

Finally, we apply the cuts $|\eta^{\text{rel}}| < 0.8$ and $|\phi^{\text{rel}}| < 0.8$ to remove a very small fraction of low-energy constituents at the periphery and use up to 128 particles per jet. The jet constituents are ordered by p_T in descending order.

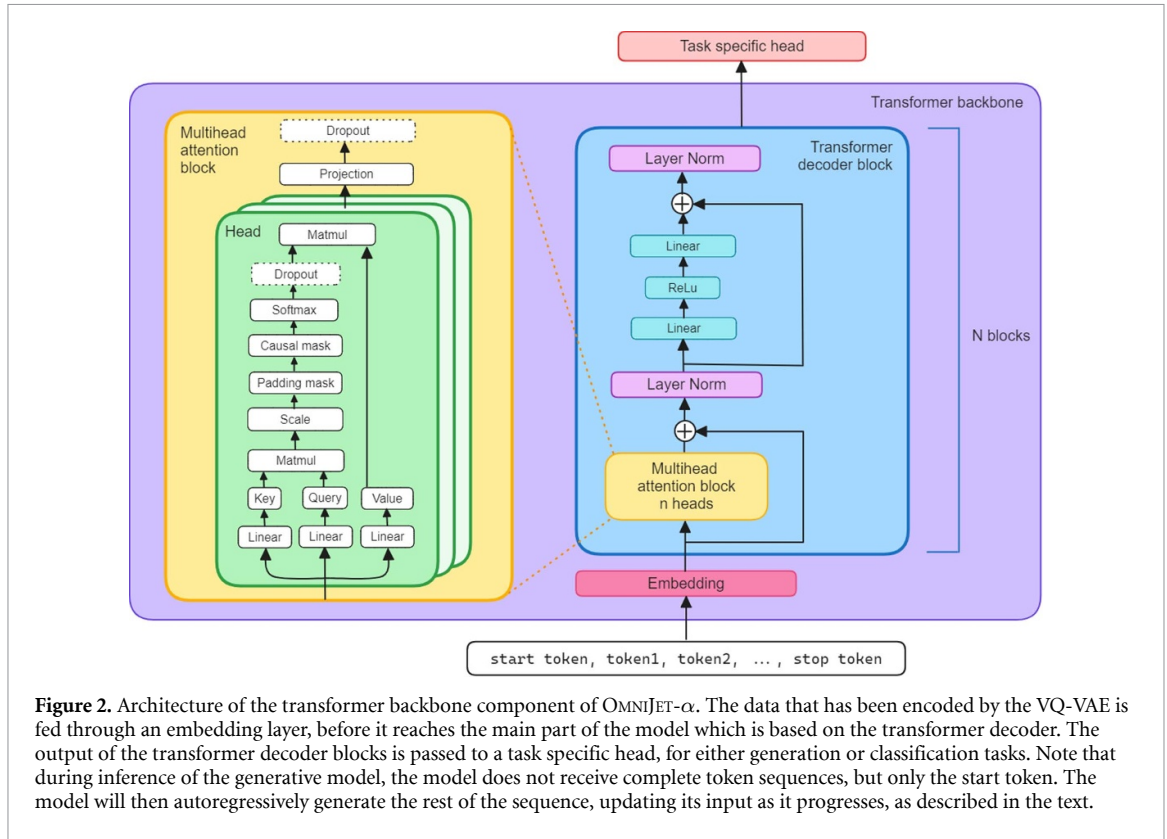
2.2. Jet constituent token creation

We explore three kinds of tokenization approaches: binned, conditional, and unconditional tokenization. In the binned approach [37], the space of input features is subdivided using a regular grid in all dimensions (e.g. a $21 \times 21 \times 21$ grid in three dimensions) and the cells in this grid are enumerated, resulting in one token per cell.

In the unconditional approach, each constituent is tokenized individually using a non-linear mapping, whereas in the conditional approach constituents are encoded and decoded conditioned on each other. We use a vector quantized variational autoencoder (VQ-VAE) [39, 46, 47, 55] to create a discrete set of jet constituent tokens both for conditional and unconditional tokenization.

The input features for the VQ-VAE are the η^{rel} , ϕ^{rel} and p_T values of the jet constituents. For the conditional tokenization, we use a transformer for both the encoder and the decoder of the VQ-VAE, whereas a simple multi-layer perceptron (MLP) is used for the unconditional tokenization. Details about the

² The difference in ϕ is signed and rectified to $-\pi$ through π . We handle those calculations using the `scikit-hep/vector` [53] and `scikit-hep/awkward` [54] packages.



different VQ-VAE models used in our studies, as well as details about the preprocessing of the input features can be found in appendix A.1.

2.3. Transformer backbone

The core of OMNIJET- α is a transformer backbone based on the GPT transformer decoder model first introduced in [56]. However, since jet constituents are permutation invariant, we do not employ the positional encoding usually used in LLMs. As input, the transformer backbone receives the generated tokens from the VQ-VAE, complemented with a start and stop token. A jet with n constituents is then represented as

$$(\text{start_token}, x_1, \dots, x_{n-1}, x_n, \text{stop_token}) \quad (3)$$

where x_i are the tokens.

The transformer backbone itself consists of an embedding layer followed by a series of GPT blocks. Each GPT block contains a multihead attention block, followed by a residual addition, layer norm [57], two linear layers with a ReLU in between, another residual addition and a final layer norm. Since this is an autoregressive model, a causal mask is passed together with the input data to the multihead attention block to prevent the model from seeing future tokens. The architecture is illustrated in figure 2.

The output from the transformer backbone is passed to a task specific head, either for generation or classification. The generative head is a single linear layer, while the classification head consists of a linear layer followed by ReLU, a sum over the constituent dimension, and a last linear layer with softmax activation function. The model is trained with $n = 8$ heads in the multi-head attention block and $N = 3$ GPT blocks. No dropout is used.

Once the generative model, i.e. the transformer backbone together with the generative head, has been trained on the tokenized data, it can generate new data autoregressively. The model has learned the probability distribution for a token x_j , given a sequence of tokens:

$$p(x_j | x_{j-1}, \dots, x_1, \text{start_token}). \quad (4)$$

The model is provided with the start token, and then samples this distribution to sequentially generate new tokens. Generation is repeated until either the stop token is generated or the maximum sequence length

(which is set to be equal to 128) is reached. The generated token sequences are then fed to the VQ-VAE decoder, which maps them into physical space for further evaluation.

The classification task can be performed either from scratch, using randomly initialized weights for both the transformer backbone and the classification head, or by fine-tuning the generative model. In the fine-tuning case, the initial weights of the transformer backbone are loaded from the generative model.

3. Results

3.1. Token quality

We first inspect how well the **tokens cover the space**. An illustration of the conditional and unconditional tokens in physical space (i.e. their corresponding η^{rel} , ϕ^{rel} and p_T values) is shown in figure 3 for the different tokenization approaches and different codebook sizes. In the unconditional case, as well as in the binning approach, the reconstruction of a token is always the same, independent of the other tokens in the jet, leading to discrete points in physical space. In the conditional case, however, the reconstruction of a token is by construction affected by the other tokens inside this jet. To visualize the spread of each conditional token in physical space, we reconstruct each token 500 times conditioned on 50 randomly chosen tokens. Each of those reconstructions is shown in the scatter plots in figure 3, where the different reconstructions of the same token are drawn in the same color. We notice that the reconstruction of each token only changes slightly when conditioned on other tokens. Thus, the 500 different reconstructions of a conditional token show up in figure 3 as a blob in physical space. This already shows that the conditional tokenization allows to cover a significantly larger volume in reconstruction space, while the unconditional tokens can only be reconstructed to distinct points in reconstruction space. We note that our approach of reconstructing each token 500 times conditioned on randomly chosen other tokens not necessarily represents the reconstructed values of actual jet constituents, as it is possible that those combinations of tokens would not appear for real jets. However, this illustrates the overall behavior of how much the reconstruction of a token can change due to the conditioning on the other tokens.

Next, we consider **distributions at the jet level** to judge the quality of the tokenization. For this and the following studies, jets are mapped into token space, and then mapped back to physical space to quantify the loss in information. Figure 4(left) shows the jet mass combined for all classes in the dataset, as was done in [39]. We observe a slight slope in the ratio plot for the unconditional tokens with codebook size 512, which is due to the underestimation of the masses (see also figure 8). The other approaches have a ratio close to 1 across the bulk of the histogram, and only show deviations at the tails where the statistics is low.

However, this inclusive distribution might hide differences at the level of individual classes and jets. We therefore also consider the difference in mass for jets before and after tokenization and reconstruction figure 4(center) for $t \rightarrow bq\bar{q}'$ jets. The unconditional tokenization leads to a systematic shift of approximately 15 GeV, while the conditional tokenization is well centered at zero. Increasing the codebook size from 512 to 8192 tokens substantially improves the resolution. This behavior is even more pronounced when considering the N -subjettiness [58] ratio τ_{32} in figure 4(right). Both conditional and unconditional tokenization with 512 tokens results in drastically shifted distributions, while the larger codebook size of 8192 recovers a peak close to zero for the mass difference. Furthermore, using 8192 tokens moves the peak of the τ_{32} difference closer to zero, while a small bias towards smaller τ_{32} in reconstructed jets is still present. In addition to that, while the mass resolution of the conditional tokens is already centered close to zero when using a codebook size of 512, the width of the distribution improves drastically from $\sigma_{512}^{\text{cond}} = 8.3$ GeV to $\sigma_{8192}^{\text{cond}} = 4.0$ GeV when moving from a codebook size of 512 to a codebook size of 8192, where σ corresponds to the standard deviation obtained from fitting a normal distribution to the mass resolution histograms. A similar behavior can be observed for other classes, where in some cases, depending on the jet observable and the jet type, the effect is even more extreme. Finally, the binning approach with a $21 \times 21 \times 21$ linear bins³ in the input features of our VQ-VAE comes close to the mass resolution of the conditional tokens with a codebook size of 8192, while the resolution of the subjettiness ratio is notably worse. Moreover, while this binning approach with 9162 tokens leads to reasonable resolution of the $t \rightarrow bq\bar{q}'$ jets shown in figure 4, we found quite drastic mismodeling for $t \rightarrow b\ell\nu$ and $H \rightarrow \ell\nu q\bar{q}'$ jets with such small codebook sizes⁴. The distributions and the

³ The binning approach with 9261 tokens in a $21 \times 21 \times 21$ grid is used as a reference, as it is the closest (but larger) number of bins (obtained with a $N \times N \times N$ grid) compared to the codebook size of 8192 tokens.

⁴ As expected, the resolution of the binning approach automatically leads to good resolution when the codebook size (i.e. the number of bins) is increased to a sufficiently large number. We found that around 640 00 tokens (corresponding to a $40 \times 40 \times 40$ grid) offer similar resolution as conditional tokenization with a codebook size of 8192.

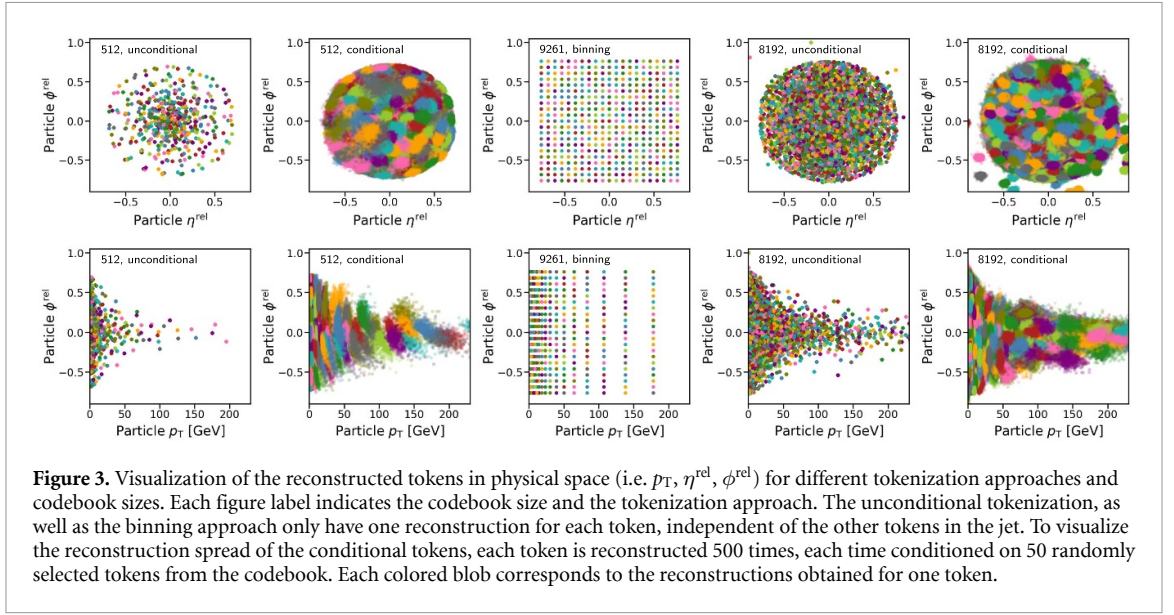


Figure 3. Visualization of the reconstructed tokens in physical space (i.e. p_T , η^{rel} , ϕ^{rel}) for different tokenization approaches and codebook sizes. Each figure label indicates the codebook size and the tokenization approach. The unconditional tokenization, as well as the binning approach only have one reconstruction for each token, independent of the other tokens in the jet. To visualize the reconstruction spread of the conditional tokens, each token is reconstructed 500 times, each time conditioned on 50 randomly selected tokens from the codebook. Each colored blob corresponds to the reconstructions obtained for one token.

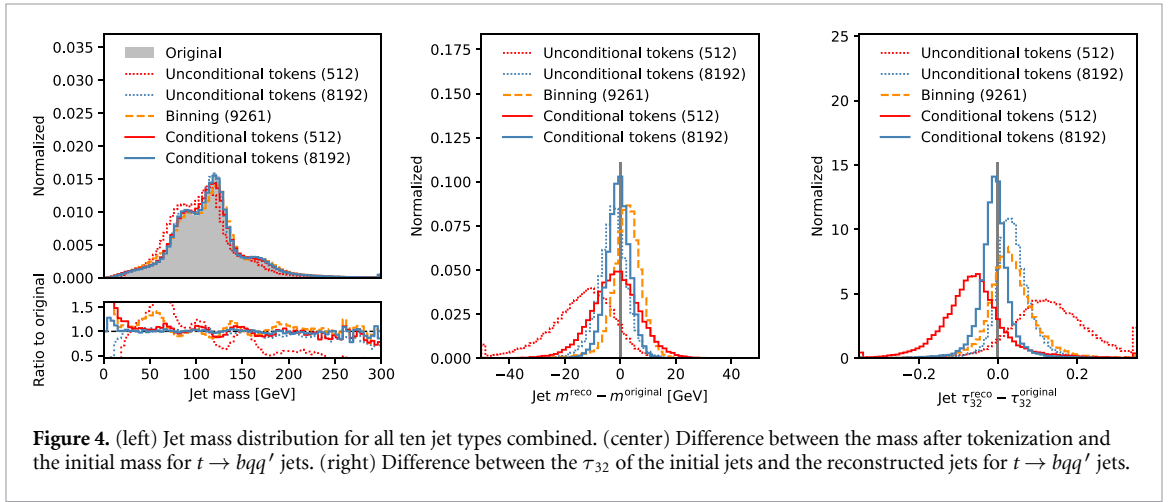


Figure 4. (left) Jet mass distribution for all ten jet types combined. (center) Difference between the mass after tokenization and the initial mass for $t \rightarrow bq\bar{q}'$ jets. (right) Difference between the τ_{32} of the initial jets and the reconstructed jets for $t \rightarrow bq\bar{q}'$ jets.

corresponding resolutions of the jet mass, jet p_T , as well as the subjettness ratios τ_{32} and τ_{21} are shown for all ten jet types individually in appendix B. Overall, the highest fidelity is achieved by conditional tokenization with a marked improvement from increasing the codebook size from 512 to 8192.

Finally, we **quantify the information loss that comes with the tokenization by training multi-class classifiers** to distinguish between the ten jet types present in the dataset. The classifiers are trained once with the original inputs, and once with the inputs after undergoing tokenization and subsequent reconstruction back into physical space. This procedure allows a direct comparison of how the loss in resolution affects reconstruction performance. We utilize two standard classifier architectures: Deep Sets [43, 59] (i.e. without particle interactions) and Transformer [36, 60] (i.e. with particle interactions) and perform this study for four different codebook sizes from 512 to 8192 tokens for the conditional tokenization approach. Details about the classifier trainings can be found in appendix A.2. Note that this approach is similar in spirit to the classifier metric proposed in [61, 62] but tests a multi-class classifier trained on these samples individually, as opposed to judging how well a classifier might distinguish original and reconstructed samples. This is necessary as e.g. points at fixed positions would be distinguishable from the original with close to perfect accuracy, rendering the test less useful.

The resulting classifier accuracy for the two different architectures is shown in figure 5. As seen in previous studies of resolution, we observe an increase of token quality as we increase the size of the VQ-VAE codebook. Furthermore, we see that the classifier performance starts to plateau with codebook sizes larger than 4096. However, even at the largest codebook size, a gap to the performance on original particles remains, motivating future work into building more accurate tokenization schemes.

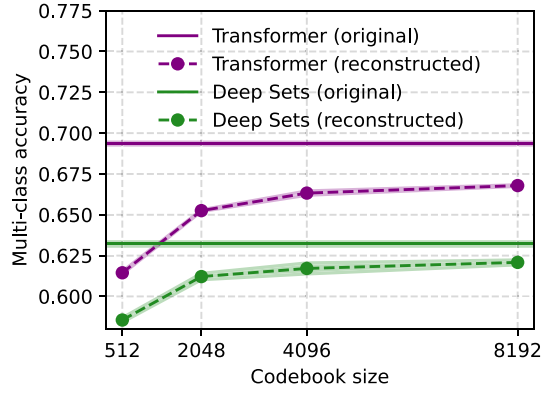


Figure 5. Token quality evaluation using a multi-class classifier. The classifier accuracy is shown for different codebook sizes and different classifier architectures (purple and green). The classifiers are also trained on the original constituents, showing an upper limit for the achievable accuracy. The reconstructed constituents are obtained using the conditional tokenization. The reported values and the uncertainty band correspond to the mean and standard deviation over 5 trainings with different random seeds for the randomly initialized weights of the classifier.

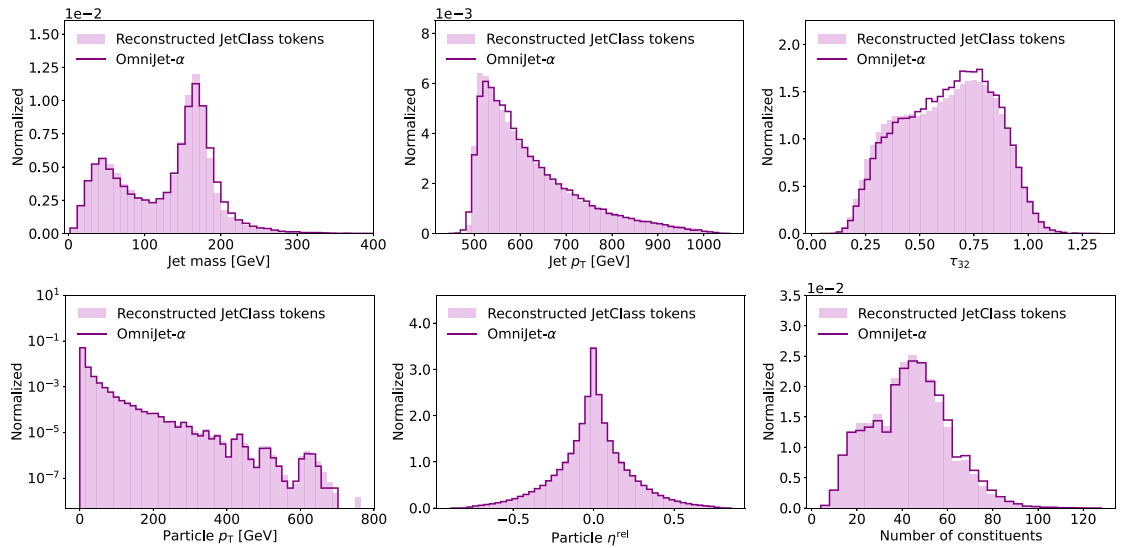


Figure 6. Comparison of generated jets from the model trained on both q/g and $t \rightarrow bqq'$ jets, to reconstructed JETCLASS tokens. The top row shows jet level distributions, while the bottom row shows distributions on the constituent level.

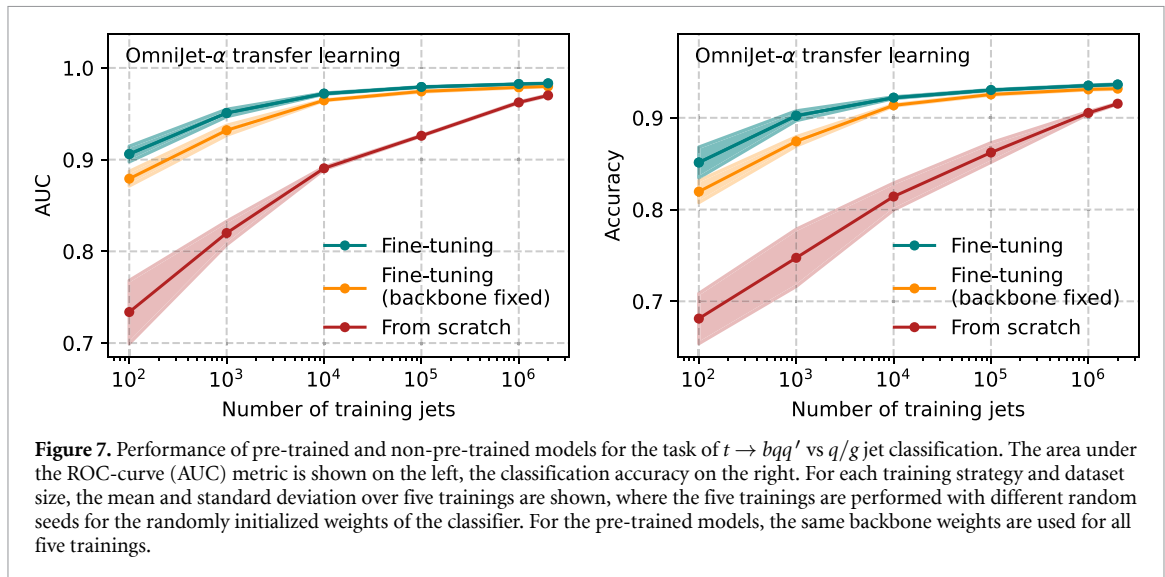
For the remaining studies we will utilize a codebook size of 8192 with conditional tokens as this leads to the overall best performance and fidelity.

3.2. Jet generation

After training the transformer backbone with the generative head, it can be used for autoregressive generation as described in section 2.3. The model was trained on three separate datasets: $t \rightarrow bqq'$ only, q/g only, and q/g and $t \rightarrow bqq'$ combined. This section will describe the combined version, since this is the model that is used for transfer learning. For a discussion of single-jet type generative results, including a comparison to the EPiC-FM model of [63], see appendix C. 480 000 events were generated from the combined model. These events contain tokens, which are then decoded back to physical space using the VQ-VAE decoder.

A comparison to reconstructed JETCLASS tokens can be seen in figure 6. We observe that in general the model is able to match the truth level tokens well. We note that the tail of the p_T spectrum of both the generated constituents and the reconstructed JETCLASS tokens contains bumps distributed around discrete values, which is consistent with our inspection of the reconstruction space shown in figure 3.

In order to quantify the performance, a classifier test (see appendix A.5 for details) is performed to distinguish generated events from reconstructed JETCLASS tokens. The test results in an AUC score of 0.54.



3.3. Transfer learning from generation to classification

To evaluate the ability of the model to generalize from generating jets to classifying them, we focus on the task of hadronic top quark tagging [7, 64], i.e. distinguishing $t \rightarrow bqq'$ and q/g jets on the JETCLASS [35] dataset. For this test, the *Next-token prediction head* is replaced by a *Classification head* while the transformer backbone is retained. We compare three training strategies: training the full architecture with randomly initialized weights (termed *from scratch*) which does not use transfer-learning and corresponds to the baseline, and two versions of fine-tuning the model obtained in the generative training step. In the *Fine-tuning* run, both the pre-trained backbone weights and the randomly initialized classification task head weights are allowed to float in the training, while in *Fine-tuning (backbone fixed)* only the classification task head is allowed to change.

The results of these training runs are presented in figure 7 as a function of the number of training examples provided to the model. We observe a significant gain in classification accuracy of both fine-tuning approaches compared to the baseline, leading to up to 17 percentage-points higher accuracy for small number of training jets, and outperforming by a few percentage-points at the highest training sample size. The difference between the two fine-tuning strategies is relatively small, with the more open training performing slightly better. Put differently, the generative pre-trained model achieves an accuracy of around 85% with 100 training examples for which the model that is trained from scratch requires more than 100 000 examples.

4. Conclusion

Foundation models for physics data are an enticing promise: trained on large amounts of data and tasks, they are expected to easily generalize to any down-stream problem, saving countless hours of human and compute time. In this paper we have taken crucial steps towards the creation of such models.

First, we expect learned representations of data to play a key role as inputs to foundation models. Representations might be continuous and rely on symmetries or learn a mapping to a discrete space as done here with tokenization. Note that while using data *raw* — i.e. without prior mapping into a representation space — might be possible when only considering a narrow range of similar datasets, it is inherently limiting when data from different sources or with different initial dimensionalities are to be considered.

Whatever representation is used, it will be important to understand and minimize the loss of information inherent in this transformation. This problem is especially important for downstream uses such as classification and regression tasks, as the loss of information can directly limit the achievable accuracy or resolution. This work introduced a set of criteria—both distribution and classifier based—that can be used to assess the quality of any representation.

Using these metrics, we found a marked increase in the resolution of relevant observables like mass and jet substructure by using a codebook size of 8192 with conditional tokenization over binning-based approaches, unconditional tokenization, and conditional tokenization with smaller codebooks. An additional classifier test further confirmed this observation.

Next, we demonstrated the training of an autoregressive generative model for jet constituents, specifically for q/g and $t \rightarrow bqq'$ jets from the JETCLASS dataset [35]. The generated distributions agree well with the ground truth, both for global jet kinematics, jet substructure, and individual constituent features. We note that while our model is the first token-based generator of JETCLASS-like examples, more extensive studies of its generative fidelity when increasing the feature-space and detailed comparison to prior non-token-based results on this dataset [63] are left for future work.

Most importantly, we report the generalization capability of OMNIJET- α from learning to generate jets in an unsupervised way, to the supervised classification between $t \rightarrow bqq'$ and q/g jets. Overall, the fine-tuned model outperformed training from scratch for all values of training examples, often by a significant margin. For example, for 1000 training jets, the fine-tuned model achieves an accuracy of approximately 90%, compared to around 75% for the freshly initialized model. While the two approaches seem to converge, even at the highest training size of 2 million jets the fine-tuned approach maintains a lead of a few percentage points. Finally, it even provides a non-trivial classification accuracy of 85%, even when trained only on 100 jets, emphasizing the value of foundation models for problems with few available labeled training examples. Since the backbone trains completely without labels, it allows us to use a large unlabeled dataset to boost the performance of a smaller labeled one. While other types of transfer have been demonstrated previously, this is the first time that the unification across the two *big* classes of tasks—classification and generation—has been achieved.

Of course, this work is only one step in building overarching foundation models. While it is the first model that achieves both classification and generation, it is not the most performant for either of these tasks. However, strategies to increase the performance exist and will be integrated. For example, the representation quality needs to be improved, possible gains from masked pre-trained have to be evaluated, architecture and training data need to be scaled up, and more extensive studies of the generalization capabilities, including training and testing on additional tasks, performed. In the medium term, strategies need to be found to align diverse datasets as well as to integrate pre-trained foundation models in community workflows. Nevertheless, the potential benefits in physics performance and compute efficiency glimpsed at in this and other works makes this a worthy endeavor.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

We thank David Shih, Michael Krämer, Michael Kagan, Frank Gaede, Sarah Heim, and Judith Katzy for stimulating discussions of foundation models for physics data and Erik Buhmann for valuable comments on the manuscript.

The authors acknowledge support by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy—EXC 2121 Quantum Universe—390833306, and under PUNCH4NFDI—Project Number 460248186. This research was supported in part through the Maxwell computational resources operated at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany.

Code availability

The code for OMNIJET- α is publicly available at https://github.com/uhh-pd-ml/omnijet_alpha.

Appendix A. Model details and hyperparameters

Different hyperparameter configurations were tested for the individual model components of OMNIJET- α . The configurations presented in the following were found to lead to stable results. However, no extensive hyperparameter optimization was performed.

A.1. VQ-VAE for token creation

Both the η^{rel} and the ϕ^{rel} values are scaled down by a factor of 3. The transverse momentum of the jet constituents is first transformed using the natural logarithm and subsequently shifted by -1.8 . The tokenization was also done without the log transform of the p_T , and was found to perform similarly. However, the logarithm transformation has the advantage that it automatically avoids negative p_T values, which is why we choose to use the log-transformed p_T . The conditional and unconditional tokenization only differ in the architecture of the encoder and decoder of the VQ-VAE.

Training for the VQ-VAE is implemented in `pytorch` [65] and `pytorch-lightning` [66].

The model architecture of the VQ-VAE encoder and decoder in the conditional tokenization approach is similar to [39] with a different set of hyperparameters. We use 4 NormFormer [60] blocks with an embedding dimension of 128 and 8 heads in the MHA for both the encoder and the decoder. We use the `vqtorch` library [55, 67] to implement the vector quantization layer with the dimension of the codebook vectors set to 4.

The mean squared error (MSE) between the tensor of the initial particle features and the reconstructed features is used as the task loss $\mathcal{L}_{\text{task}}$. The total loss is then set to

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \alpha \cdot \mathcal{L}_{\text{commit}} \quad (\text{A1})$$

with $\alpha = 10$. An affine transformation is used for a joint transformation of all codes and dead codes are replaced with a frequency of 10 iterations. The parameter β which trades off the importance of updating the embeddings from the encoder z_e and the code vectors z_q is set to $\beta = 0.9$. Lastly, we use a synchronized update rule [55, 67] with $\nu = 1$.

In the unconditional approach, we use the same hyperparameters as outlined above, with the only difference that the architecture of the encoder and decoder is a simple MLP with 3 hidden layers of dimension 128 and ReLU activation function.

All VQ-VAE models are trained on all 10 classes of the JETCLASS dataset [35].

A.2. Classifiers for token quality evaluation

The DeepSets [43, 59] classifier consists of a per-particle MLP Φ with shared weights across all particles inside the jet with 3 hidden layers of dimension 100, 100 and 256. The output of the network Φ is then aggregated with a sum and fed into another MLP with 3 hidden layers of dimension 100 followed by a 10-dimension output layer with softmax activation function.

The Transformer classifier consists of 5 NormFormer [60] blocks, followed by two class-attention blocks with a class token as query, inspired by the ParT [10] architecture. The output of the last class-attention block is fed into a MLP with two hidden layers of dimension 128, followed by a softmaxed 10-dimensional output layer.

The classifiers are trained with the AdamW [68] optimizer with a maximum learning rate of 0.005 (0.001) for the DeepSets (Transformer) classifier and weight decay 0.01. The learning rate first linearly increased from 0.002 (0.0005) during the first 4 training epochs, after which it is linearly decreased to the initial learning rate over 20 epochs and then linearly decreased to a final learning rate of 0.001 (0.0003), following the one-cycle learning rate schedule [69].

The classifiers for those token quality evaluations are trained on 10M jets from the JETCLASS dataset [35].

A.3. Transformer backbone

When training the transformer backbone, cross entropy is used as a loss function and Adam [70] with a learning rate of 0.001 as optimizer. The model had access to 10M $t \rightarrow bqq'$ jet events and 10M q/g jet events. Note that this means that the model trained on these two jet types combined had access to twice as much data. All versions were trained for 30 epochs, and the model state with the lowest validation loss was chosen for the further analysis.

A.4. Transfer learning

To perform the transfer learning from the generative task to the classification task, we change the head of the OMNIJET- α model to the classification head and load the weights of the backbone trained for the generative task. We explore two variations of fine-tuning the pre-trained backbone to the classification task: training all weights of the model with the same learning rate (referred to as *Fine-tuning* in section 3.3) and keeping the weights of the backbone fixed at the state obtained from the generative model (referred to as *Fine-tuning (backbone fixed)* in section 3.3). For the *From scratch* trainings we start the training with randomly initialized weights of the whole model. The training is performed with the AdamW [68] optimizer with a constant learning rate of 0.0001 and weight decay 0.01. The transfer learning studies are performed on the test dataset from the default train-val-test split of the JetClass dataset, which includes 2M jets per jet-type, leading to 4M jets in total for our studies with q/g and $t \rightarrow bqq'$ jets. We split those 4M jets into 2M jets for training, and 1M jets for validation and testing, respectively. Since those trainings, depending on the size of the training dataset, tend to show overfitting quite quickly, we stop those trainings when the validation loss does not improve for multiple epochs. The threshold of this early stopping is adjusted to the training dataset size, with a patience of 20 epochs for a training dataset of 100, 1000 and 10 000 jets, a patience of 10 epochs for a training dataset size of 100 000 and 1000 000, and a patience of five for trainings with 2000 000 training jets. For each training dataset size we run five trainings with different random seeds and the epoch with the smallest validation loss is chosen for evaluation.

A.5. Classifier tests

In order to quantify the performance of the generative model, a classifier test using the structure of the DeepSets classifier from appendix A.2 is performed. In this case however, the three hidden layers of the particle MLP Φ all have dimension 10.

A number of 48 000 generated events are combined with equally many reconstructed tokens from the test set of JETCLASS. The two datasets are combined and shuffled, and a train/val/test split of 0.6/0.2/0.2 is used. The model is trained for 100 epochs with binary cross entropy loss and Adam [70] with learning rate 0.001 as optimizer. The model state with the lowest validation loss is chosen for evaluation. The resulting AUC scores are 0.54 for the model trained on q/g and $t \rightarrow bqq'$ jets combined and 0.57 for the ones trained on single-type jets.

Appendix B. Token quality

Additional plots of the jet mass, the jet transverse momentum, as well as the subjettness ratios are shown in figures 8–15.

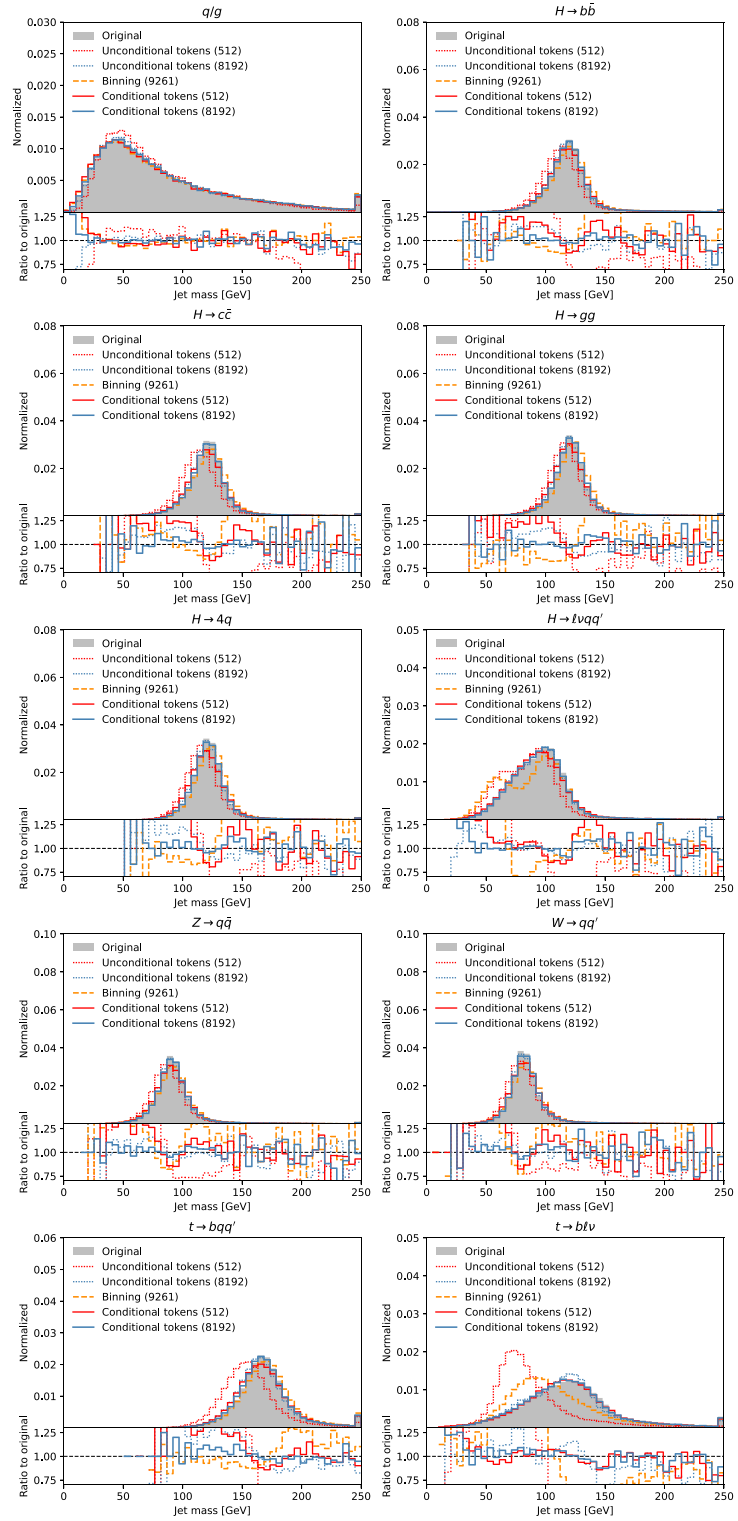


Figure 8. Jet mass distribution for different tokenization approaches and codebook sizes.

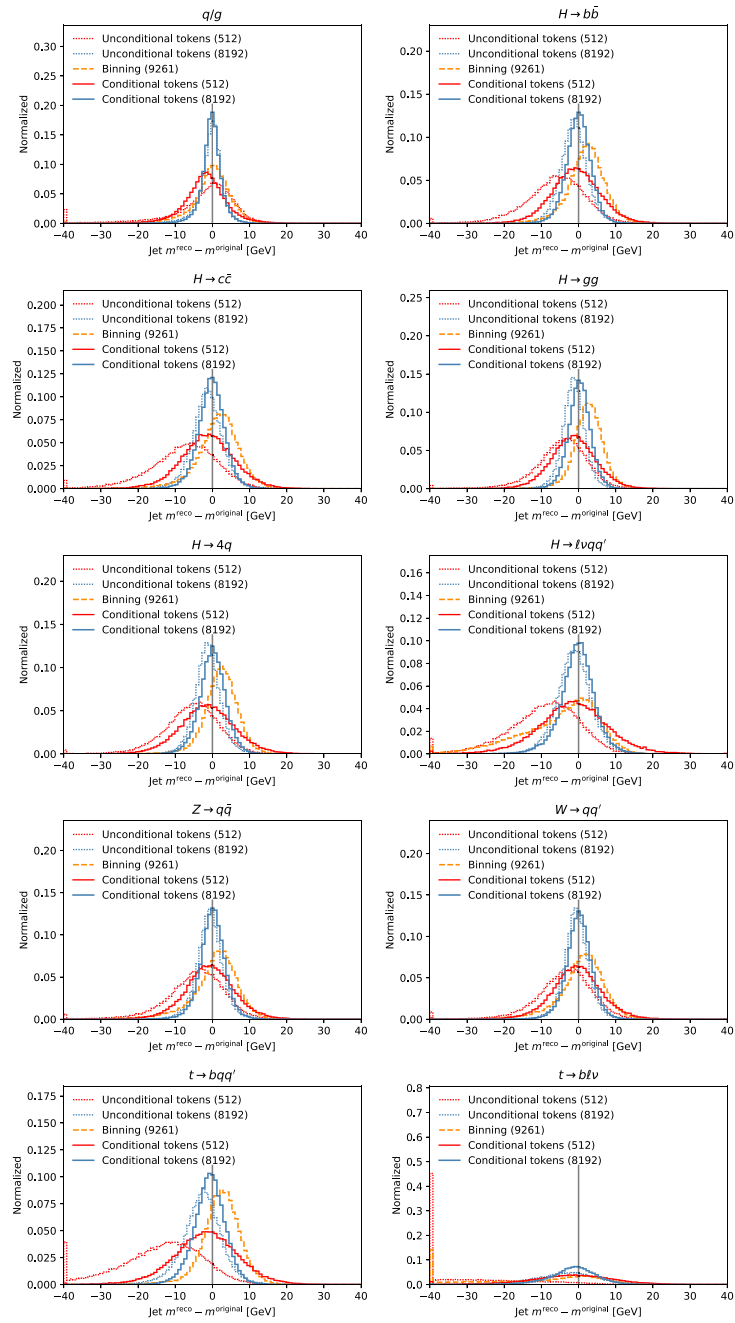


Figure 9. Jet mass resolution for different tokenization approaches and codebook sizes.

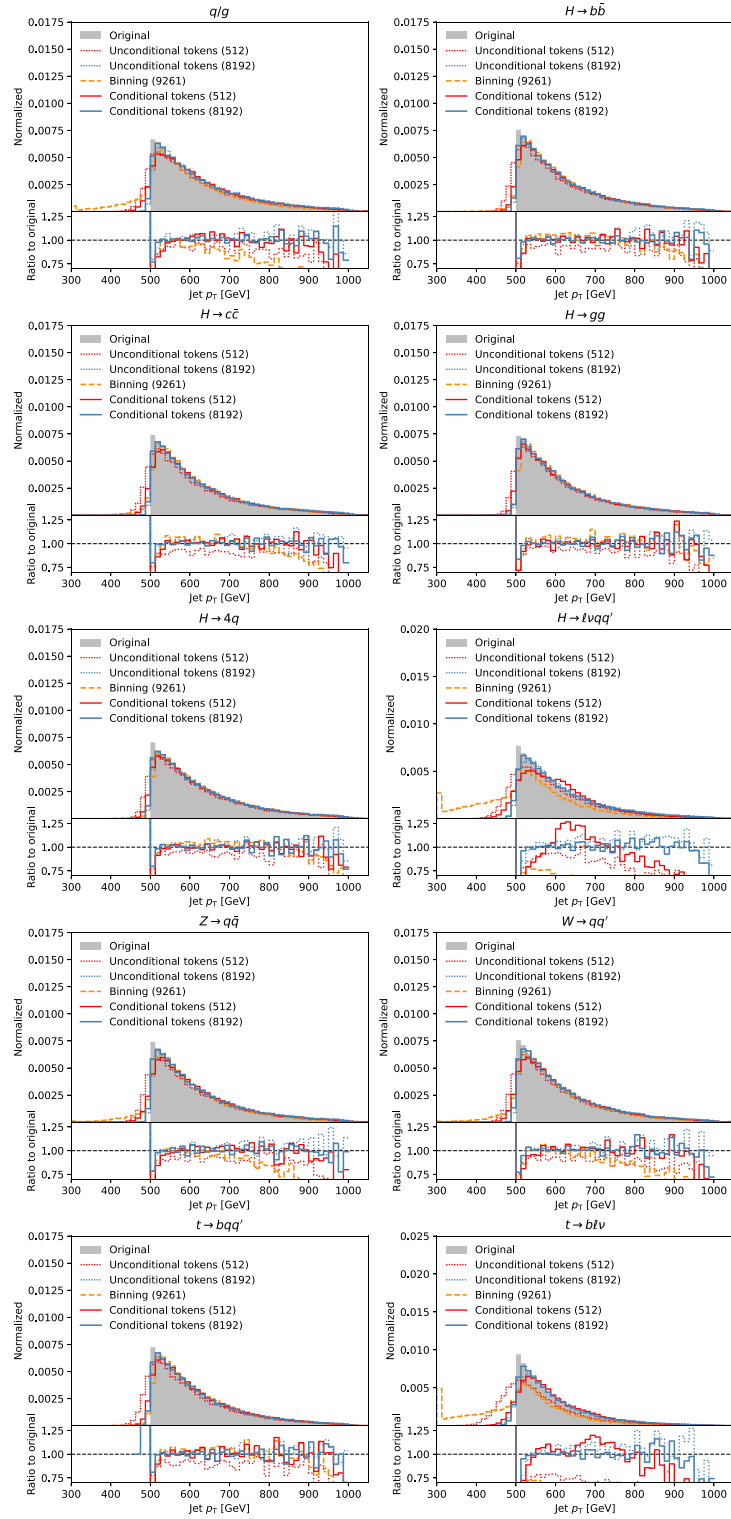


Figure 10. Jet p_T distribution for different tokenization approaches and codebook sizes.

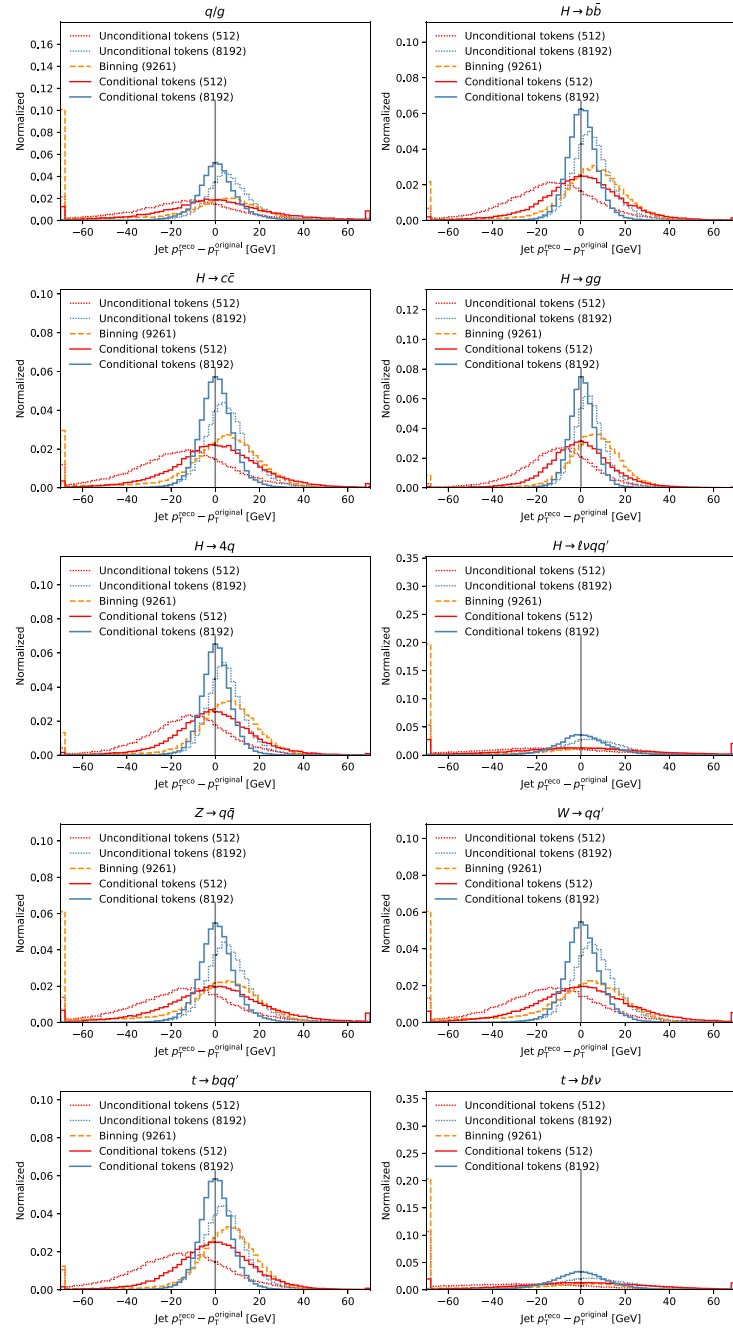


Figure 11. Jet p_T resolution for different tokenization approaches and codebook sizes.

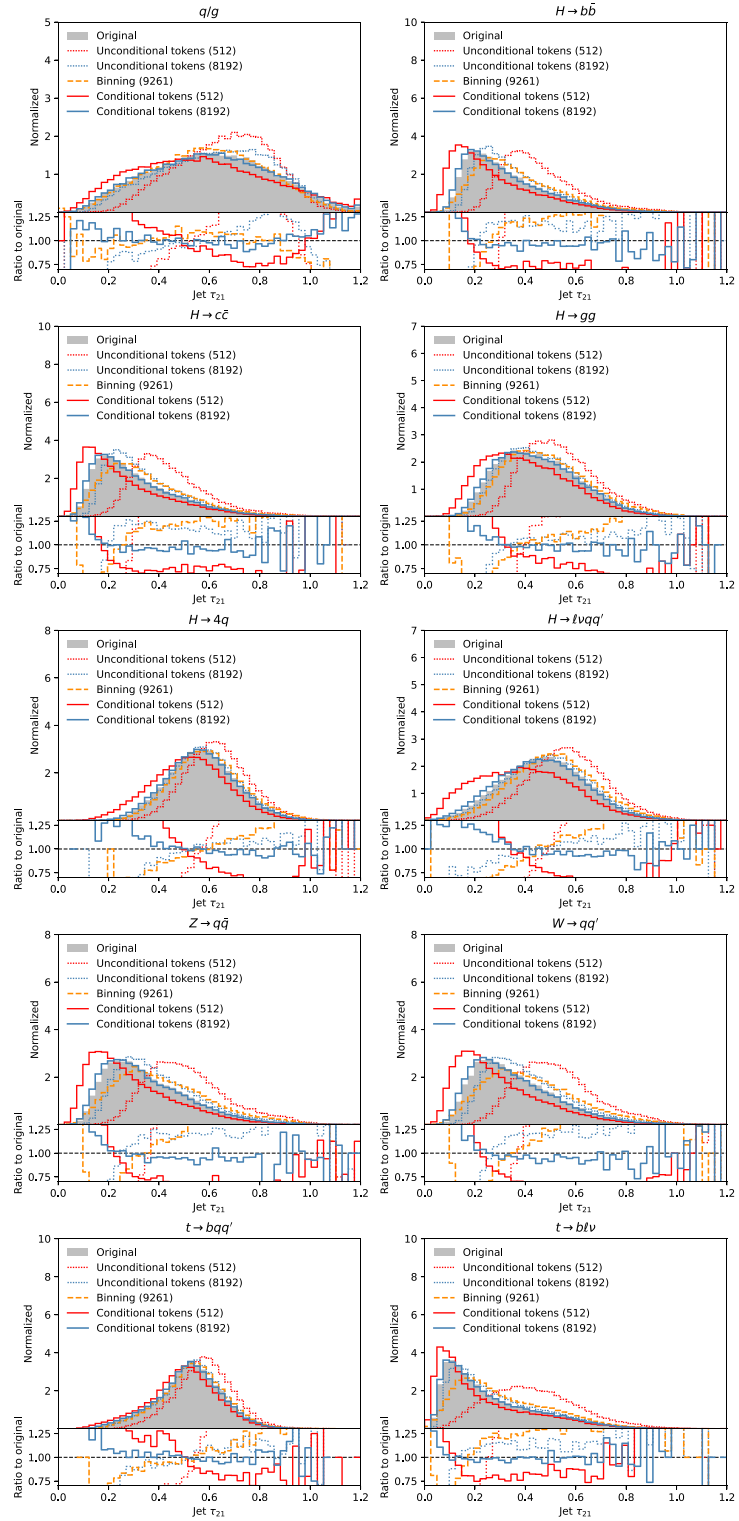


Figure 12. Jet τ_{21} distribution for different tokenization approaches and codebook sizes.

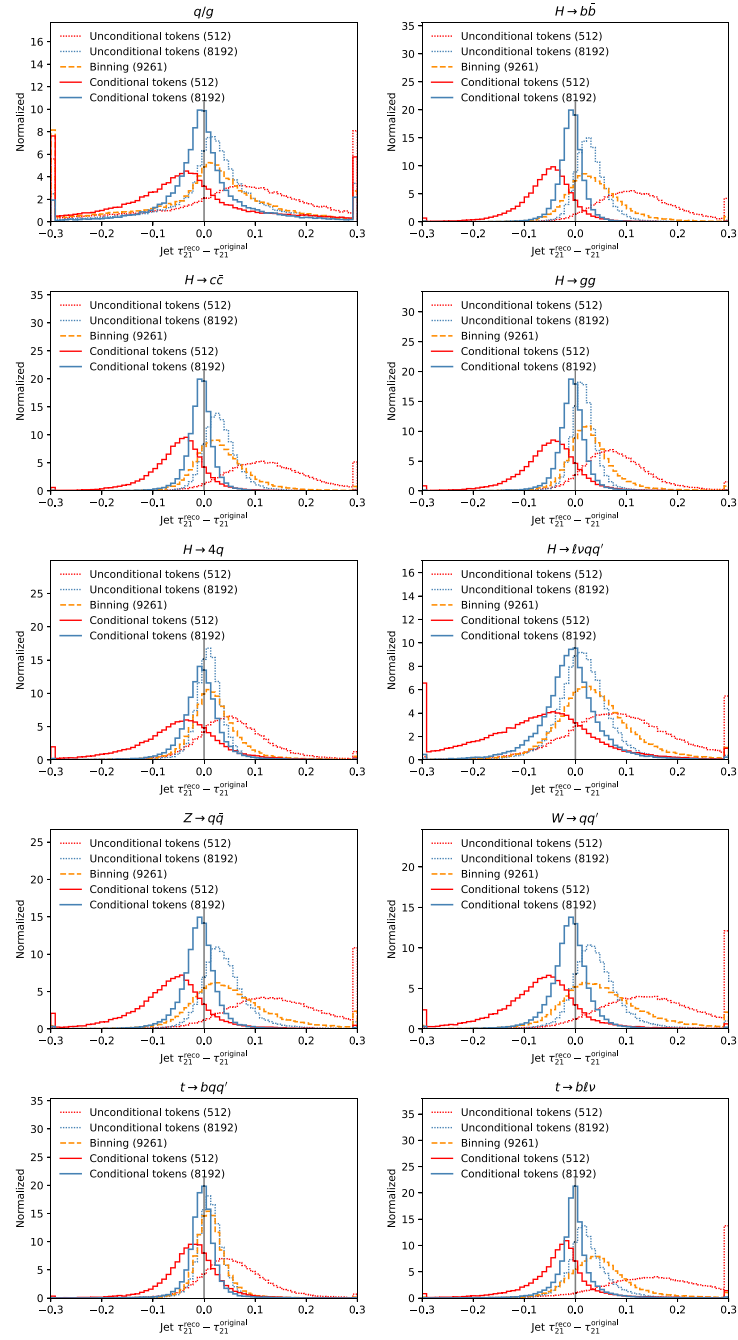


Figure 13. Jet τ_{21} resolution for different tokenization approaches and codebook sizes.

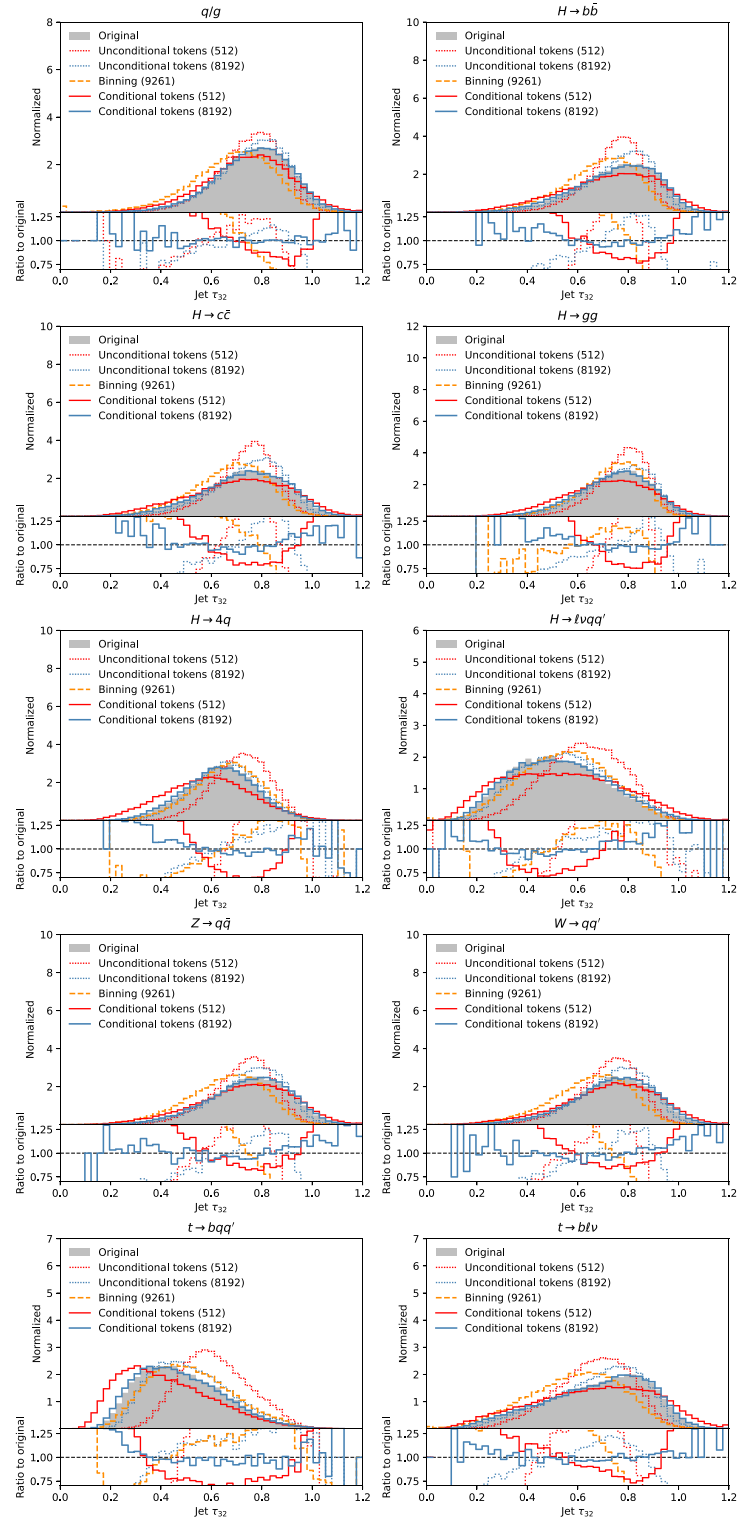


Figure 14. Jet τ_{32} distribution for different tokenization approaches and codebook sizes.

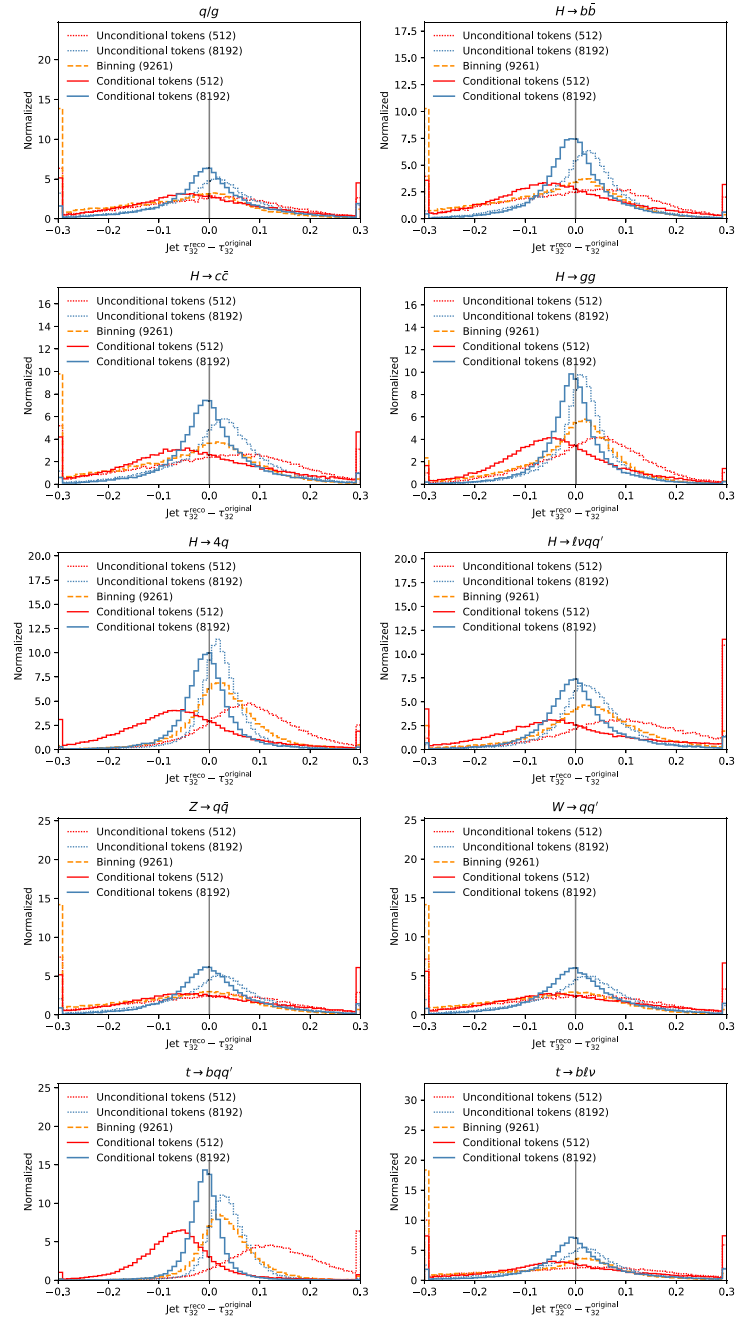
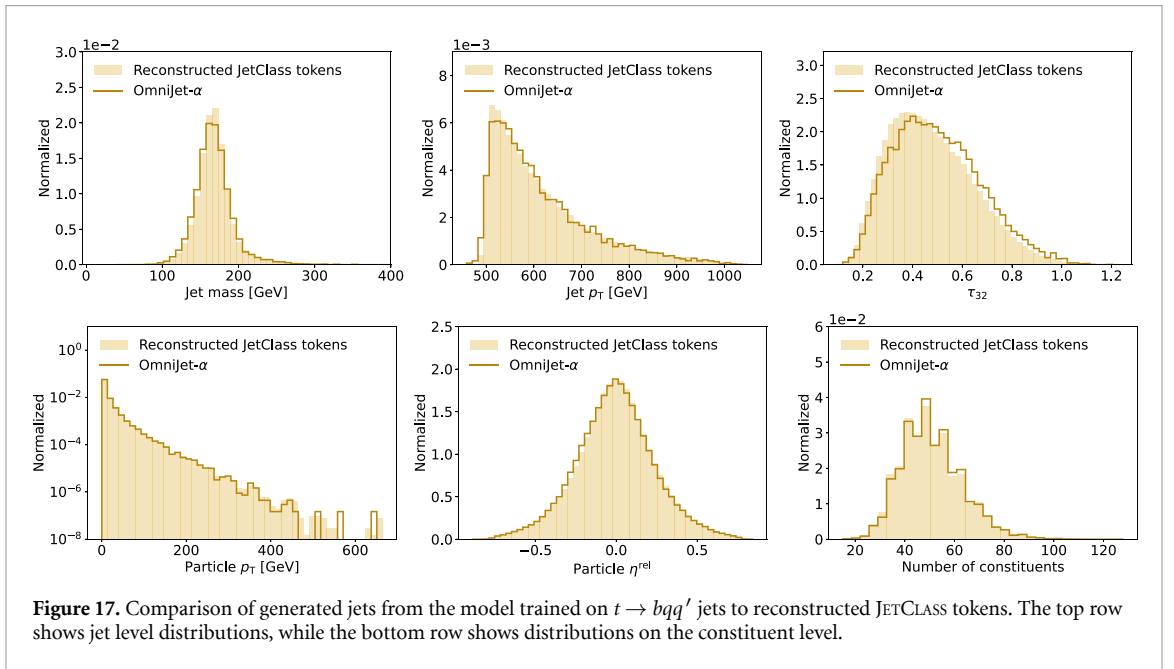
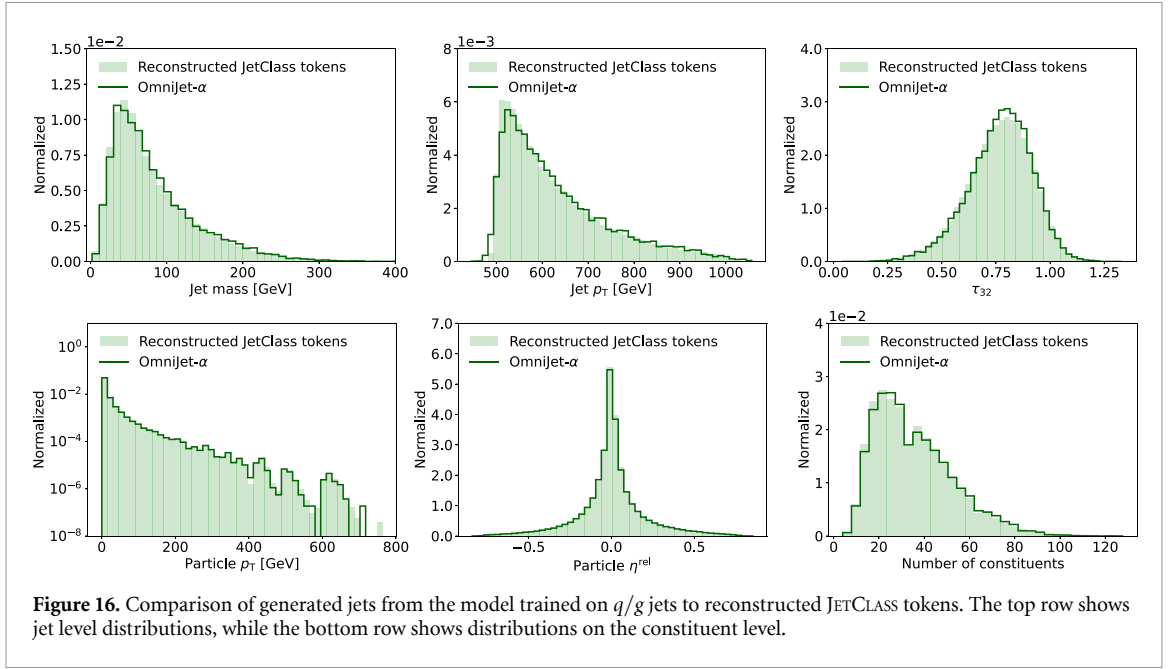


Figure 15. Jet τ_{32} resolution for different tokenization approaches and codebook sizes.

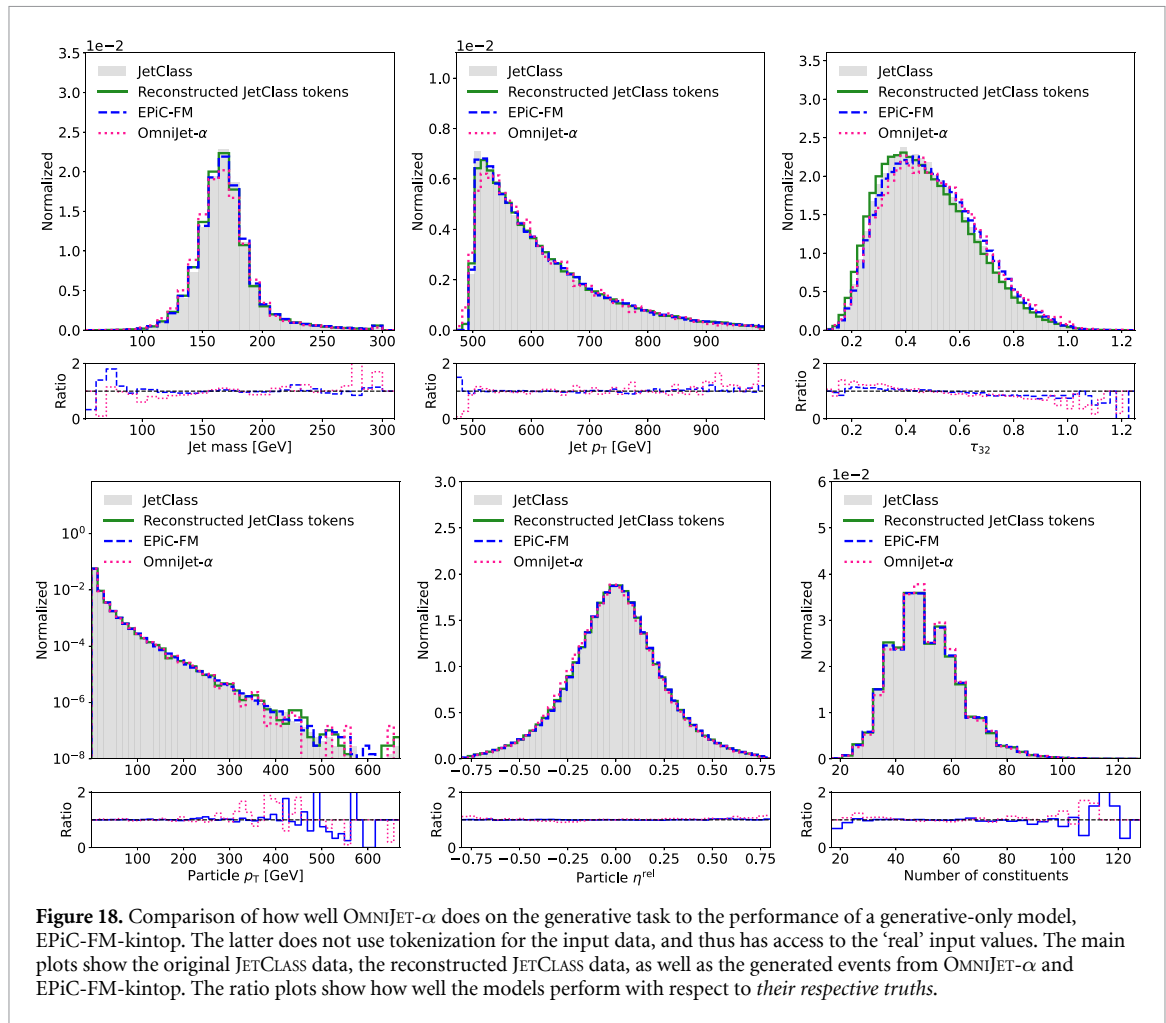


Appendix C. Generative model trained on single-jet data

To test the generative performance, the generative model was also trained on single-jet type data — 10M jets each of $t \rightarrow bqq'$ and q/g — separately. For these training, no tests of the task-transfer to classification were performed.

The result of the q/g jet training is shown in figure 16, of the $t \rightarrow bqq'$ jet training in figure 17. In the q/g case, we see a good agreement between the reconstructed tokens and the generated events. However, it seems to be somewhat more difficult for the model to accurately model τ_{32} for $t \rightarrow bqq'$ jets, which is also mirrored for this quantity in the combined model (see figure 6).

It is interesting to compare the result of OMNIJET- α with that of a different generative model. EPiC-FM [63] was the first generative model trained on the JETCLASS dataset, utilizing flow matching and operating without tokenization. The result of the comparison can be seen in figure 18. The plots show the JETCLASS data, the reconstructed JETCLASS token from this work, the EPiC-FM-kintop generated events, and the OMNIJET- α generated events. We use the more challenging $t \rightarrow bqq'$ class for comparison.



The ratio plots under the main plots show the generated events compared to *their respective truths*: direct JETCLASS for EPiC-FM-kintop and Reconstructed JETCLASS tokens for OMNIJET- α . Hence, the ratios show how well the respective generative models learn to replicate their training data. In general, we see that both models are doing well. OMNIJET- α has a somewhat higher discrepancy in the tails of all distributions except for constituent η^{rel} and the number of constituents. The most difficult distribution is the constituent p_T , with bumps in the tail, which could also be seen in figure 17.

ORCID iDs

Joschka Birk <https://orcid.org/0000-0002-1931-0127>

Anna Hallin <https://orcid.org/0000-0002-1551-814X>

Gregor Kasieczka <https://orcid.org/0000-0003-3457-2755>

References

- [1] Bommasani R *et al* 2022 On the opportunities and risks of foundation models (arXiv:2108.07258 [cs.LG])
- [2] Devlin J, Chang M-W, Lee K and Toutanova K 2019 BERT: pre-training of deep bidirectional transformers for language understanding (arXiv:1810.04805 [cs.CL])
- [3] Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V and Zettlemoyer L 2019 BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension (arXiv:1910.13461 [cs.CL])
- [4] Brown T B *et al* 2020 Language models are few-shot learners (arXiv:2005.14165 [cs.CL])
- [5] Touvron H *et al* 2023 LLaMA: open and efficient foundation language models (arXiv:2302.13971 [cs.CL])
- [6] Ramesh A, Pavlov M, Goh G, Gray S, Voss C, Radford A, Chen M and Sutskever I 2022 Zero-shot text-to-image generation (arXiv:2102.12092 [cs.CV])
- [7] Kasieczka G *et al* 2019 The machine learning landscape of top taggers *SciPost Phys.* **7** 014
- [8] Karagiorgi G, Kasieczka G, Kravitz S, Nachman B and Shih D 2022 Machine learning in the search for new fundamental physics *Nat. Rev. Phys.* **4** 399–412
- [9] Macaluso S and Shih D 2018 Pulling out all the tops with computer vision and deep learning *J. High Energy Phys.* **JHEP10(2018)121**
- [10] Qu H, Li C and Qian S 2022 Particle transformer for jet tagging *Proc. 39th Int. Conf. on Machine Learning* pp 18281–92

- [11] Vigl M, Hartman N and Heinrich L 2024 Finetuning foundation models for joint analysis optimization (arXiv:2401.13536 [hep-ex])
- [12] Albrecht J *et al* (HEP Software Foundation) 2019 A roadmap for HEP software and computing R&D for the 2020s *Comput. Softw. Big Sci.* **3** 7
- [13] Boehnlein A *et al* 2022 *HL-LHC Software and Computing Review Panel Report* (available at: <http://cds.cern.ch/record/2803119/>)
- [14] Paganini M, de Oliveira L and Nachman B 2018 Accelerating science with generative adversarial networks: an application to 3D particle showers in multilayer calorimeters *Phys. Rev. Lett.* **120** 042003
- [15] Buhmann E, Diefenbacher S, Eren E, Gaede F, Kasieczka G, Korol A and Krüger K 2021 Getting high: high fidelity simulation of high granularity calorimeters with high speed *Comput. Softw. Big Sci.* **5** 13
- [16] Buhmann E, Gaede F, Kasieczka G, Korol A, Korcari W, Krüger K and McKeown P 2023 CaloClouds II: ultra-fast geometry-independent highly-granular calorimeter simulation (arXiv:2309.05704 [physics.ins-det])
- [17] Adelmann A *et al* 2022 New directions for surrogate models and differentiable programming for high energy physics detector simulation *Snowmass 2021*
- [18] Butter A *et al* 2023 Machine learning and LHC event generation *SciPost Phys.* **14** 079
- [19] Hashemi H and Krause C 2023 Deep generative models for detector signature simulation: an analytical taxonomy (arXiv:2312.09597 [physics.ins-det])
- [20] Butter A, Plehn T and Winterhalder R 2019 How to GAN LHC events *SciPost Phys.* **7** 075
- [21] de Oliveira L, Paganini M and Nachman B 2017 Learning particle physics by example: location-aware generative adversarial networks for physics synthesis *Comput. Softw. Big Sci.* **1** 4
- [22] Jack Y *et al* 2024 Les Houches guide to reusable ML models in LHC analyses (arXiv:2312.14575 [hep-ph])
- [23] Bieringer S, Kasieczka G, Kieseler J and Trabs M 2024 Classifier surrogates: sharing AI-based searches with the world (arXiv:2402.15558 [hep-ph])
- [24] Dillon B M, Kasieczka G, Olischlager H, Plehn T, Sorrenson P and Vogel L 2022 Symmetries, safety and self-supervision *SciPost Phys.* **12** 188
- [25] Favaro L, Krämer M, Modak T, Plehn T and Rüschkamp J 2023 Semi-visible jets, energy-based models, and self-supervision (arXiv:2312.03067 [hep-ph])
- [26] Dillon B M, Favaro L, Feiden F, Modak T and Plehn T 2023 Anomalies, representations, and self-supervision (arXiv:2301.04660 [hep-ph])
- [27] Park S E, Harris P and Ostdiek B 2023 Neural embedding: learning the embedding of the manifold of physics data *J. High Energy Phys.* **JHEP07(2023)108**
- [28] Dillon B M, Mastandrea R and Nachman B 2022 Self-supervised anomaly detection for new physics *Phys. Rev. D* **106** 056005
- [29] Benato L *et al* 2022 Shared data and algorithms for deep learning in fundamental physics *Comput. Softw. Big Sci.* **6** 9
- [30] Liu J, Ghosh A, Smith D, Baldi P and Whiteson D 2023 Generalizing to new geometries with geometry-aware autoregressive models (GAAMs) for fast calorimeter simulation *J. Instrum.* **18** 11003
- [31] Salamani D, Zaborowska A and Pokorski W 2023 MetaHEP: meta learning for fast shower simulation of high energy physics experiments *Phys. Lett. B* **844** 138079
- [32] Dolan M J and Ore A 2022 Metalearning and data augmentation for mass-generalized jet taggers *Phys. Rev. D* **105** 094030
- [33] Beauchesne H, Chen Z-E and Chiang C-W 2024 Improving the performance of weak supervision searches using transfer and meta-learning *J. High Energy Phys.* **JHEP02(2024)138**
- [34] Kasieczka G, Plehn T, Thompson J and Russel M 2019 Top quark tagging reference dataset (Zenodo) (available at: <https://doi.org/10.5281/zenodo.2603256>)
- [35] Qu H, Li C and Qian S 2022 JetClass: a large-scale dataset for deep learning in jet physics (Zenodo) (available at: <https://doi.org/10.5281/zenodo.6619768>)
- [36] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L and Polosukhin I 2017 Attention is all you need *31st Int. Conf. on Neural Information Processing Systems*
- [37] Finke T, Krämer M, Mück A and Tönshoff J 2023 Learning the language of QCD jets with transformers *J. High Energy Phys.* **JHEP06(2023)184**
- [38] Butter A, Huetsch N, Palacios Schweitzer S, Plehn T, Sorrenson P and Spinner J 2023 Jet diffusion versus JetGPT—modern networks for the LHC (arXiv:2305.10475 [hep-ph])
- [39] Heinrich L, Golling T, Kagan M, Klein S, Leigh M, Osadchy M and Raine J A 2024 Masked particle modeling on sets: towards self-supervised high energy physics foundation models (arXiv:2401.13537 [hep-ph])
- [40] Huang A, Melkani Y, Calafiura P, Lazar A, Murnane D T, Pham M-T and Ju X 2024 A language model for particle tracking *Connecting The Dots 2023*
- [41] Hashemi B, Hartmann N, Sharifzadeh S, Kahn J and Kuhr T 2023 Ultra-high-resolution detector simulation with intra-event aware GAN and self-supervised relational reasoning (arXiv:2303.08046 [physics.ins-det])
- [42] Harris P, Kagan M, Krupa J, Maier B and Woodward N 2024 Re-simulation-based self-supervised learning for pre-training foundation models (arXiv:2403.07066 [hep-ph])
- [43] Komiske P T, Metodieff E M and Thaler J 2019 Energy flow networks: deep sets for particle jets *J. High Energy Phys.* **JHEP01(2019)121**
- [44] Buhmann E, Kasieczka G and Thaler J 2023 EpiC-GAN: equivariant point cloud generation for particle jets (arXiv:2301.08128 [hep-ph])
- [45] Buhmann E, Diefenbacher S, Eren E, Gaede F, Kasieczka G, Korol A, Korcari W, Krüger K and McKeown P 2023 CaloClouds: fast geometry-independent highly-granular calorimeter simulation *J. Instrum.* **18** 11025
- [46] van den Oord A, Vinyals O and Kavukcuoglu K 2018 Neural discrete representation learning (arXiv:1711.00937 [cs.LG])
- [47] Bao H, Dong Li, Piao S and Wei F 2022 BEiT: BERT pre-training of image transformers (arXiv:2106.08254 [cs.CV])
- [48] Alwall J, Frederix R, Frixione S, Hirschi V, Maltoni F, Mattelaer O, Shao H-S, Stelzer T, Torrielli P and Zaro M 2014 The automated computation of tree-level and next-to-leading order differential cross sections and their matching to parton shower simulations *J. High Energy Phys.* **JHEP07(2014)079**
- [49] Sjöstrand T, Ask S, Christiansen J R, Corke R, Desai N, Ilten P, Mrenna S, Prestel S, Rasmussen C O and Skands P Z 2015 An introduction to PYTHIA 8.2 *Comput. Phys. Commun.* **191** 159–77
- [50] de Favereau J, Delaere C, Demin P, Giammanco A, Lemaitre V, Mertens A and Selvaggi M 2014 DELPHES 3: a modular framework for fast simulation of a generic collider experiment *J. High Energy Phys.* **JHEP02(2014)057**
- [51] The CMS Collaboration 2008 The CMS experiment at the CERN LHC *J. Instrum.* **3** S08004

- [52] Cacciari M, Salam G P and Soyez G 2008 The anti- k_t jet clustering algorithm *J. High Energy Phys.* [JHEP04\(2008\)063](#)
- [53] Schreiner H, Pivarski J and Chopra S 2023 vector (Zenodo) (available at: <https://doi.org/10.5281/zenodo.7671687>)
- [54] Pivarski J, Osborne I, Ifrim I, Schreiner H, Hollands A, Biswas A, Das P, Roy Choudhury S, Smith N and Goyal M 2024 Awkward Array (Zenodo) (available at: <https://doi.org/10.5281/zenodo.10498548>)
- [55] Huh M, Cheung B, Agrawal P and Isola P 2023 Straightening out the straight-through estimator: overcoming optimization challenges in vector quantized networks (arXiv:2305.08842 [cs.LG])
- [56] Radford A, Narasimhan K, Salimans T and Sutskever I 2018 Improving language understanding by generative pre-training (available at: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)
- [57] Ba J L, Kiros J R and Hinton G E 2016 Layer normalization (arXiv:1607.06450 [stat.ML])
- [58] Thaler J and Van Tilburg K 2011 Identifying boosted objects with N -subjettiness *J. High Energy Phys.* [JHEP03\(2011\)015](#)
- [59] Zaheer M, Kottur S, Ravanbakhsh S, Poczos B Salakhutdinov R and Smola A 2018 Deep sets (arXiv:1703.06114 [cs.LG])
- [60] Shleifer S, Weston J and Ott M 2021 Normformer: improved transformer pretraining with extra normalization (arXiv:2110.09456 [cs.CL])
- [61] Krause C and Shih D 2023 Fast and accurate simulations of calorimeter showers with normalizing flows *Phys. Rev. D* **107** 113003
- [62] Das R, Favaro L, Heimerl T, Krause C, Plehn T and Shih D 2023 How to understand limitations of generative networks (arXiv:2305.16774 [hep-ph])
- [63] Birk J, Buhmann E, Ewen C, Kasieczka G and Shih D 2023 Flow matching beyond kinematics: generating jets with particle-id and trajectory displacement information (arXiv:2312.00123 [hep-ph])
- [64] Kasieczka G, Plehn T, Russell M and Schell T 2017 Deep-learning top taggers or the end of QCD? *J. High Energy Phys.* [JHEP05\(2017\)006](#)
- [65] Paszke A et al 2019 PyTorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* vol 32, ed H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox and R Garnett (Curran Associates, Inc.) pp 8024–35
- [66] Falcon W (The PyTorch Lightning Team) 2024 Pytorch lightning (Zenodo) (available at: <https://doi.org/10.5281/zenodo.10779019>)
- [67] Huh M 2022 vqtorch: PyTorch package for vector quantization (available at: <https://github.com/minyoungg/vqtorch>)
- [68] Loshchilov I and Hutter F 2019 Decoupled weight decay regularization (arXiv:1711.05101 [cs.LG])
- [69] Smith L N 2018 A disciplined approach to neural network hyper-parameters: Part 1—learning rate, batch size, momentum, and weight decay (arXiv:1803.09820 [cs.LG])
- [70] Kingma D P and Ba J 2017 Adam: a method for stochastic optimization (arXiv:1412.6980 [cs.LG])



DATASET ARTICLE

OPEN ACCESS

RECEIVED
22 January 2025REVISED
13 May 2025ACCEPTED FOR PUBLICATION
17 June 2025PUBLISHED
7 July 2025

Original Content from
this work may be used
under the terms of the
Creative Commons
Attribution 4.0 licence.

Any further distribution
of this work must
maintain attribution to
the author(s) and the title
of the work, journal
citation and DOI.



Aspen Open Jets: unlocking LHC data for foundation models in particle physics

Oz Amram¹ , Luca Anzalone^{2,3} , Joschka Birk⁴ , Darius A Faroughy⁵ , Anna Hallin^{4,*} ,
Gregor Kasieczka^{4,6} , Michael Krämer⁷ , Ian Pang⁵ , Humberto Reyes-Gonzalez⁷ and David Shih⁵

¹ Fermi National Accelerator Laboratory, Batavia, IL 60510, United States of America

² Department of Physics and Astronomy (DIFA), University of Bologna, 40127 Bologna, Italy

³ Istituto Nazionale di Fisica Nucleare (INFN), 40127 Bologna, Italy

⁴ Institut für Experimentalphysik, Universität Hamburg, 22761 Hamburg, Germany

⁵ NHETC, Dept. of Physics and Astronomy, Rutgers University, Piscataway, NJ 08854, United States of America

⁶ Center for Data and Computing in Natural Sciences (CDCS), 22607 Hamburg, Germany

⁷ Institut für Theoretische Teilchenphysik und Kosmologie, RWTH Aachen University, 52074 Aachen, Germany

* Author to whom any correspondence should be addressed.

E-mail: anna.hallin@uni-hamburg.de, oamram@fnal.gov, luca.anzalone2@unibo.it, joschka.birk@uni-hamburg.de,
darius.faroughy@rutgers.edu, gregor.kasieczka@uni-hamburg.de, mkraemer@physik.rwth-aachen.de,
ian.pang@physics.rutgers.edu, humberto.reyes@rwth-aachen.de and shih@physics.rutgers.edu

Keywords: foundation models, particle physics, transfer learning, generation, jet physics, transformer, open data

Abstract

Foundation models are deep learning models pre-trained on large amounts of data which are capable of generalizing to multiple datasets and/or downstream tasks. This work demonstrates how data collected by the CMS experiment at the Large Hadron Collider can be useful in pre-training foundation models for HEP. Specifically, we introduce the ASPENOPENJETS (AOJs) dataset, consisting of approximately 178 M high p_T jets derived from CMS 2016 Open Data. We show how pre-training the OMNIJET- α foundation model on AOJs improves performance on generative tasks with significant domain shift: generating boosted top and QCD jets from the simulated JetClass dataset. In addition to demonstrating the power of pre-training of a jet-based foundation model on actual proton–proton collision data, we provide the ML-ready derived AOJs dataset for further public use.

1. Introduction

While particle physics has long used machine learning techniques and is leading the way in adopting modern deep learning methods, the development and application of powerful foundation models—which have transformed natural language processing and computer vision—is still in its early stages. For the purposes of this work, a foundation model will be a model that has been pre-trained on a large amount of data for a specific task, and that can be used for different tasks downstream [1]. The pre-training thus serves as a foundation, upon which the downstream tasks can be built. Well-known foundation models include large language models such as GPT-3 [2], BERT [3] and LLaMA [4], image models such as Imagen [5], and mixed-modality models such as CLIP [6] and ALIGN [7].

There are several advantages to the use of foundation models. One example is the potential to boost the performance on small datasets. Having access to a model pre-trained on a larger, similar dataset provides a ‘head start’ for the downstream task, allowing the model to focus on the intricacies of the smaller dataset rather than having to rediscover the general structure of the data. In addition, a foundation model can save both human and computational resources. While pre-training may be a resource-intensive task, the downstream models would require less training, less data, and less time spent on optimization compared to what would be required if these models were built from scratch. Another interesting aspect is that a trained foundation model could enable implicit sharing of data and resources between different parts of the research community, a sharing that would not otherwise be possible.

Foundation models are poised to play a central role in particle physics, especially in the context of experiments at the Large Hadron Collider (LHC). They are able to process the huge amounts of complex and

diverse data generated by the LHC and have the potential to integrate multiple modalities—such as images, high-level event features or particle four-vectors—allowing more comprehensive data analyses. Current work on foundation models for particle physics has mainly focused on different event generation and classification tasks [8–13], investigating pre-training strategies, encoding schemes and different architectures, and quantifying the transfer learning and generalization capabilities of these models.

In this paper, we explore the potential of using large quantities of Open Data from the LHC for pre-training foundation models. The LHC experiments have made significant amounts of data publicly available [14–18]. Foundation models can leverage the LHC open data for training and fine-tuning, potentially integrating different data modalities and data from different experiments. Furthermore, training foundation models on real data will result in a more accurate representation of particle physics processes than training on simulations. Finally, foundation models trained on a large variety of actual LHC data will help to promote transparency and collaboration in particle physics research.

Here we present the ASPENOPENJETS (AOJs) dataset and study the transfer learning of foundation models obtained with this dataset. The AOJ dataset has been constructed from the CMS 2016 ‘JetHT’ open data [16, 17] and consists of 178 M jets, making it the largest ML-ready dataset for LHC applications. Jets are important objects at the LHC, consisting of collimated sprays of particles. Previous open data studies of CMS jets have explored the substructure and energy flow of the jets [19–21] and the usefulness of open data for quark/gluon tagging [22], but this is the first study to prepare and make public a form of the CMS open data that allows easy use of the dataset for phenomenological studies, especially for training machine learning models.

As an initial proof-of-concept study, we pre-train the OMNIJET- α [11] foundation model on the AOJ dataset, which is expected to mostly contain QCD jets, and then fine-tune it to generate jets of different types from the JetClass [23] dataset of simulated jets. We do two such fine-tunings: first for JetClass QCD jets, which occupy a different kinematic region compared to the jets in AOJ, and then for top jets, which differ significantly in several features. We show that pre-training on AOJ indeed benefits the downstream task, making it easier for the model to generate these types of jets with fewer training examples compared to training from scratch.

The utility of CMS open data for pre-training has been previously explored in [8], using 1 M single-lepton events at the object level which were collected by CMS in 2015. Here we enlarge the scope and complexity by several orders of magnitude to encompass 178 M jet events at the full constituent level, that is, including not only the high-level features of the jets themselves, but also the features of all the particles inside the jet. Furthermore, the authors of [8] used event selections to shape the open data into a similar topology to the intended target simulated data. In our work, no such shaping is needed.

This paper is organized as follows. Section 2 describes the CMS open data and the method used to derive AOJ. It also includes a brief description of JetClass, which will be used for the downstream task. Our application example is described in section 3, detailing the method and results. Finally, we present our conclusions in section 4. Links to the codebase and the AOJ dataset itself can be found after the conclusions.

2. Data

2.1. CMS open data

The CMS experiment [24] is a large general-purpose detector at the LHC. The CMS collaboration has recently released 16.4 fb^{-1} of data from their 2016 runs (G and H) [16, 17] and a corresponding set of simulations. In contrast to previous data releases [14, 15, 25], this data has been provided in the simplified MINIAOD format [26], as well as the even further simplified NANOAOD format [27]. In these simplified analysis formats, the central CMS reconstruction steps have been applied, discarding much of the low-level content of the event in favor of a simplified description of the high-level objects (jets, electrons, muons, etc) commonly used in data analysis. These formats are therefore significantly easier to use for those outside of the collaboration.

The inner layers of CMS consist of a silicon pixel and silicon strip tracker used to measure charged particle trajectories. These are followed by a lead tungstate crystal electromagnetic calorimeter and a brass and scintillator hadronic calorimeter. These detector elements are enclosed in a large solenoid, which produces a magnetic field of 3.8 T. Muon stations are located outside the iron return yoke of the magnet. Events of interest are selected using a two-stage trigger system. The first stage (L1) is implemented in custom hardware to select potentially interesting events within a fixed latency of about $4 \mu\text{s}$ [28] using information from the calorimeters and muon stations. Events accepted by the L1 system are sent to the high-level trigger (HLT), which consists of a dedicated computing cluster which runs a speed-optimized version of the full reconstruction. The HLT uses this more accurate reconstruction of the event to select events to be saved at a rate of approximately 1 kHz [29].

Table 1. Breakdown of the expected number of jet types in the AOJ dataset from MC simulations.

Jet type	Approx. Number
QCD	1.8×10^8
W	6.3×10^5
Z	1.8×10^5
Top	1.3×10^5
$H \rightarrow b\bar{b}$	500

CMS uses a particle-flow algorithm [30], which combines the information from the different detector systems to reconstruct and identify each individual particle in an event. Particles are classified as either muons, electrons, photons, charged hadrons or neutral hadrons. Jets are clustered from the particle-flow candidates in an event using the anti- k_t jet finding algorithm [31, 32]. The typical large-radius jet size used in CMS for jet substructure searches and measurements is $R = 0.8$, commonly referred to as AK8 jets. For AK8 jets the pileup-per-particle identification algorithm (PUPPI) [33, 34] is used to mitigate the effect of pileup. The PUPPI algorithm assigns a weight to each particle based on the probability that it originates from the primary vertex or from pileup. The momentum of AK8 jets is determined from the weighted vector sum of all particles inside the jet. Additional corrections are applied based on simulation and data-driven measurements to better calibrate the measured jet momentum to the true momentum of the particle from the hard process [35].

2.2. Extracting AOJs

The NANOAOJ format does not natively store information of the jet constituents. However, an extended version (PFNANO), which does include this information, can be easily produced from MINIAOD using tools provided by CMS [36]. We therefore, for all data and simulations samples used, start with the MINIAOD format, process it to PFNANO and then apply our selections to select the jets of interest for our dataset.

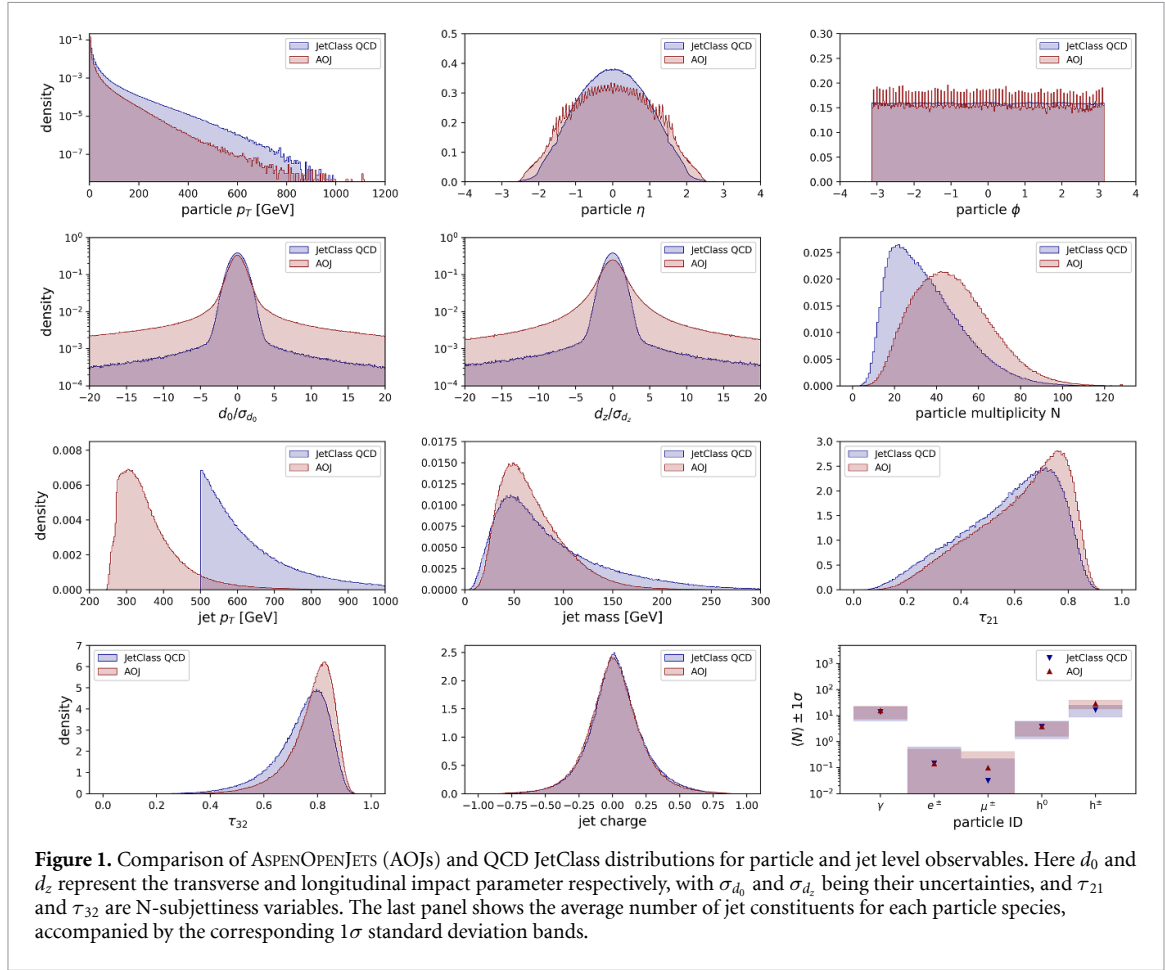
In order to construct the largest sample of jets possible, we consider all events from the ‘JetHT’ [16, 17] Run-2 dataset released by CMS to this date. The ‘JetHT’ dataset includes all events that have been selected by one of the HLT triggers related to jet momenta or total event hadronic activity (HT). Events are required to either have a high scalar sum of jet transverse momenta, or contain at least one or two jets with high energy. For the exact details of all different trigger paths used, see [16, 17]. Because we do not specify a specific trigger, the sample consists of events from an inclusive ‘or’ of any of the available triggers of this dataset. We note that this choice works well for building a large dataset for the purposes of foundation model training, but may introduce subtle kinematic biases resulting from the combined thresholds of the different triggers, which would complicate its use in a physics-based analysis.

Events are required to pass several data-quality filters commonly used within CMS to remove events that are likely to cause problems in the reconstruction. AK8 jets are required to have transverse momentum greater than 300 GeV⁸. The jet p_T spectrum is steeply falling, and the 300 GeV threshold was chosen to balance having a large dataset of jets for the pre-training task with the consideration of matching the high-momentum kinematic regime typical of typical jet substructure studies. Jets are also required to have an absolute value of pseudorapidity (η) less than 2.5, so that they fall within the range of the CMS tracker. The AK8 jets are also required to pass standard CMS quality criteria used to reject leptons reconstructed as jets and jets that may be poorly reconstructed. We construct our sample out of all AK8 jets that pass these criteria, with no limitation on the number of selected jets per event.

Applying these selections to the CMS open data yields a dataset of 178 million jets, which we call the AOJs. Due to the large cross section of QCD processes, the vast majority of these jets are expected to originate from light quarks and gluons. A small fraction of the jets are expected to originate from the decays of boosted heavy resonances. A rough estimate of the number of boosted fully merged jets produced by W , Z , top, and Higgs bosons in the $b\bar{b}$ decay mode are estimated with Monte Carlo (MC) simulations provided by CMS [37], with the remainder assumed to originate from QCD. The numbers are reported in table 1. More details on the MC samples used are given in [appendix](#).

For each jet we store its transverse momentum (p_T), pseudorapidity (η), and azimuthal angular coordinate (ϕ). We also store its mass, groomed with the softdrop algorithm [38] as computed within the CMS reconstruction. Up to 150 constituents of the jet are stored. For each constituent, its 4-momentum is stored in the format (p_x, p_y, p_z, E) . We additionally store its transverse impact parameter (d_0) and

⁸ The jet transverse momentum (p_T) distribution in figure 1 extends down to 250 GeV as it is computed directly from vector sum of the jet constituents. Whereas, 300 GeV threshold is applied to the jet after additional corrections have been applied to the jet energy.



longitudinal impact parameter (d_z) with their uncertainties, the charge of the candidate, its particle-ID (PID) in the PDG format⁹ [39] and its weight from the PUPPI algorithm. We also include additional jet substructure quantities computed within the CMS reconstruction, including the number of constituents in the jet, N -subjettiness variables [40], various jet-tagging observables from the CMS implementation of ParticleNet [41, 42] and a regression of the jet mass from ParticleNet [42].

In figure 1 we show the distributions (red histograms) of several particle and jet-level features, based on a sub-sample of 700 K jets. Interestingly, the effects of detector granularity are seen as spikes in the particle η and ϕ distributions. In figure 2 we plot a 2D histogram in the $\eta - \phi$ plane. A closer look reveals a grid-like structure corresponding to the granularity of the detector. Furthermore, there are vertical features indicating the endcaps at $|\eta| \sim 1.6$ and several ‘dead cells’ throughout the detector. The distribution of the subjettiness ratio τ_{32} is consistent with the expectation that the AOJ is predominantly composed of QCD jets. The last panel in figure 1 shows the average number of constituents per jet for each PID, accompanied by the corresponding 1σ standard deviation bands.

2.3. JetClass

The JetClass [23] dataset is based on simulations and was first introduced in [9]. It contains both particle-level and jet-level information for 125 M simulated jets, distributed over 10 different jet types and divided into a training (100 M), validation (5 M) and test (20 M) set. The simulation is performed with MADGRAPH5_AMC@NLO [43] for the hard process, followed by PYTHIA [44] for parton showering and hadronization, and DELPHES [45] with the CMS detector [24] card for the detector response. Jets are required to have a transverse momentum of $500 \text{ GeV} < p_T < 1000 \text{ GeV}$ and a pseudorapidity $\eta < 2$, which is more restrictive than the criteria used for AOJ. We overlay the JetClass QCD particle- and jet-level distributions (blue histograms) in figure 1 to facilitate easy comparison with the corresponding AOJ distributions.

⁹ Note that neutral hadrons (h^0) are assigned the PID = 130 of the neutral kaon K_L^0 while positively/negatively charged hadrons ($h^\pm$) are assigned PID = ± 211 of the charged pion.

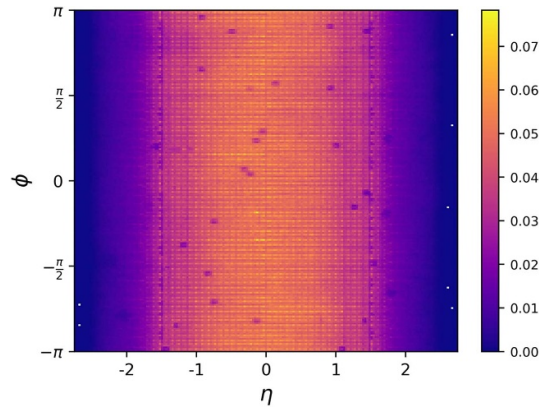


Figure 2. The particle-level positional distribution of the ASPENOPENJETS in the $\eta - \phi$ plane, illustrating the granularity of the CMS detector. Note the appearance of vertical features indicating the endcaps at $|\eta| \sim 1.6$ and multiple ‘dead cells’ throughout the detector.

3. Experiment

In this section, we demonstrate the usefulness of the AOJ dataset in ML-related applications. In particular, we pre-train a foundation model on the full AOJ dataset.

3.1. Method

The foundation model chosen for this work is OMNIJET- α [11], a GPT-style [46] model that predicts the next token in a sequence. The features of the jet constituents¹⁰ are first converted into integer tokens using a VQ-VAE [10, 47–49] model, representing jets as sequences of tokens. The backbone of OMNIJET- α is trained to predict the next token in these sequences. After pre-training, the model can generate new sequences autoregressively, which are then decoded by the frozen VQ-VAE back into physical space. Additionally, the pre-trained backbone weights can be loaded and used for other tasks, such as jet classification. In this work, we focus on dataset transfer learning for jet generation and leave the exploration of task switching for future studies.

The features used for training are the constituent kinematics $(p_T, \Delta\eta, \Delta\phi)$, where $\Delta\eta = \eta - \eta_{\text{jet}}$ and $\Delta\phi = \phi - \phi_{\text{jet}}$ represent the relative coordinates with respect to the jet axis. Before tokenization, these features were preprocessed using standardization¹¹. The AOJ dataset was tokenized using a codebook size of 8192 tokens, employing VQ-VAE hyperparameters as outlined in [11], with two modifications: the number of attention heads was reduced from 8 to 4 without sacrificing performance, and the AdamW optimizer was replaced with the Ranger optimizer [50]. A larger codebook with ~ 32 K tokens was also tested but produced results comparable to the 8 K configuration. Therefore, we only present results for the 8 K tokenization. The backbone model was trained on the full 178 M AOJ dataset with hyperparameters as specified in [11]. The model was trained for 900 K steps on 12 GPUs with a batch size of 256 per GPU, and the final state at the end of the training was selected as the backbone model.

To evaluate the transfer learning capabilities of OMNIJET- α , we devised a downstream task to test its ability to generate jets of a different type than those it was pre-trained on, and to adapt to datasets with different kinematic cuts. The fine-tuning process involved loading the pre-trained backbone weights and retraining it using jets from the JetClass [23] dataset. To systematically evaluate the performance of the model, we trained it on six datasets with $10^2, 10^3, 10^4, 10^5, 10^6, 10^7$ jets to study the effect of dataset size D , and fine-tuned the model separately on QCD jets (q/g) and top-quark jets ($t \rightarrow bqq'$) to evaluate its adaptability to different jet substructure types. For each configuration, the model with the lowest validation loss during fine-tuning was selected for jet generation. To provide a baseline for comparison, we used a separate VQ-VAE tokenizer and trained generative models with the same backbone architecture from scratch using randomly initialized weights. This VQ-VAE is the one from [11] that was trained on all 10 classes of the JetClass dataset. The generative models were trained on the tokenized QCD and top-quark samples. This approach allowed us to quantify the performance improvement achieved by pre-training on the AOJ dataset.

¹⁰ The jet constituents are p_T ordered and up to the first 128 constituents are considered when training the models.

¹¹ Standardization is a common technique used to bring feature values into a manageable range for neural networks. By subtracting the mean and dividing by the standard deviation, the data is transformed to having a mean of zero and standard deviation of one.

Both training strategies used a batch size of 256¹² per GPU¹³ and a patience of 10 epochs. For training samples with sizes up to 100 K jets, the validation set was the same size as the training set. For larger samples, we used validation set sizes of 200 K jets for 1 M training samples and 500 K jets for 10 M training samples.

3.2. Performance metrics

We consider two complementary evaluation metrics that compare the 1D distribution of a high-level observable $\mathcal{O}$ between the generated jets and the target JetClass reference data:

- Kullback–Leibler divergence:

$$\text{KL}^{\mathcal{O}}(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right). \quad (1)$$

The probability densities $P(x)$ and $Q(x)$ for the observable $\mathcal{O}$ are obtained from the normalized histogram bin counts and $\text{KL}^{\mathcal{O}}(P||Q)$ is computed using `scipy.stats.entropy` [51]. Note that $\text{KL}^{\mathcal{O}}(P||Q)$ is anti-symmetric. In our results, we take P to be the generated distribution and Q to be the reference JetClass distribution.

- Wasserstein-1 distance:

$$W_1^{\mathcal{O}}(P, Q) = \min_{\pi \in \Pi} \sum_{x, y} |x - y| \pi(x, y). \quad (2)$$

Here $\Pi(P, Q)$ is the space of all joint ‘coupling’ distributions whose marginals are the observable $\mathcal{O}$ distributions P and Q . This quantity is computed with `scipy.stats.wasserstein_distance` [51]. Unlike the $\text{KL}^{\mathcal{O}}(P||Q)$ divergence, $W_1^{\mathcal{O}}(P, Q)$ is a symmetric similarity metric based on optimal transport, i.e. the optimal cost plan that morphs P into Q .

While further detailed performance evaluations such as classifier tests [52–54] could also be envisioned, this paper is first and foremost demonstrating a use case for the AOJ dataset in the context of foundation models. As such, we are rather concerned with the overall qualitative behavior, for which the 1D distributions suffice.

3.3. Results

A comparison of the two training strategies –‘fine-tuned’ and ‘from scratch’– is presented in figure 3 for QCD and in figure 4 for top-quark jets. The rows show various high-level distributions, including jet p_T , jet mass, subjeiness ratios τ_{21} and τ_{32} , and particle multiplicity N (i.e. the number of constituents in a jet). The left-most column in each figure displays the distributions generated by the fine-tuned models, while the second column shows the distributions generated by the models trained from scratch. Each plot also includes the reference JetClass distribution (200 K reference jets) as a filled histogram and, for additional context, the AOJ distributions (dotted lines) in the left-most column.

Overall, we find that the pre-trained models are able to achieve a performance gain when using a fraction of the total fine-tuning data. This can be seen in the third and last columns of figures 3 and 4, where the fine-tuned models (blue lines) typically achieve better¹⁴ metric scores than the models trained from scratch (red lines). This is apparent for the KL divergence and Wasserstein-1 metrics across different training sample sizes below 1 M jets. However, as the available training data exceeds 1 M jets, we find that the fine-tuning begins to saturate, and the fine-tuned and from-scratch models perform similarly within uncertainties. In some features, the fine-tuned models even exhibit a slight decrease in performance compared to those trained entirely from scratch, indicating a diminishing return from pre-training at larger dataset scales. The reasons for this are unclear and would require further study. One possibility is that the AOJ tokenization is suboptimal for describing JetClass and this is exposed with enough JetClass training data. Another possibility is that the pre-training itself can harm model performance by locking in the weights to a suboptimal part of parameter space, a phenomenon that has been observed previously in the ML literature [55] and was termed ‘ossification’.

¹² For training samples of 100 jets, the batch size was reduced to 100.

¹³ We used a single GPU to train the models with training sample sizes less than or equal to 100 K, and multiple GPUs for models trained on more data.

¹⁴ However, there are instances where models trained from scratch perform equally well on a single metric within uncertainties. For example, this occurs with KL^{p_T} for training sample size $D = 10^3$. This behavior can be attributed to the definition of KL divergence, which penalizes high-probability regions in the generated data where the true probability distribution is low. Therefore, it is crucial to evaluate performance using multiple metrics to gain a comprehensive understanding.

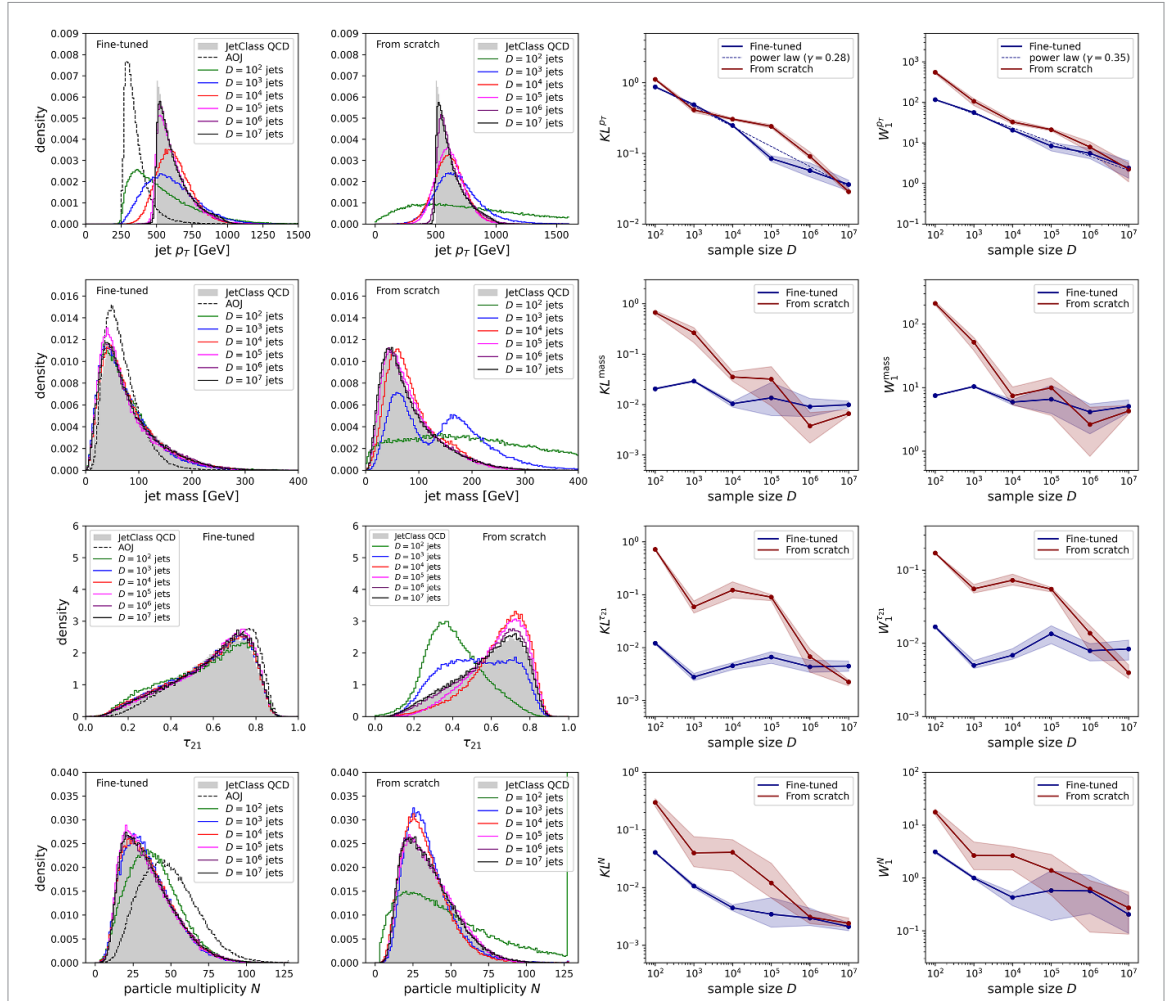


Figure 3. A comparison of the generation quality of models trained on QCD jets from JetClass, for different training sample sizes D . The performance of a foundation model pre-trained on AOJ (far left) is compared to a model trained from scratch (center left). The generation quality is compared across several high level features of the jets. Two quantitative metrics, the Kullback–Leibler divergence (center right) and Wasserstein-1 distance (far right), are computed as a function of the training sample size D to compare how well the generated jets from each model matches the target distribution from JetClass. We also report the mean value and the envelopes over 5 trainings with different random seeds. When applicable, we also show power-law fits $\propto D^{-\gamma}$ to the metrics.

We now examine in more detail the transfer learning from AOJ to JetClass for QCD jets, as shown in figure 3. The primary challenge in fine-tuning arises from the differences in the domains of the jet p_T distributions and particle multiplicities between the two datasets. JetClass imposes a narrower p_T window between 500 GeV and 1000 GeV and tends to have around ~ 30 particles per jet on average, while AOJ only applies a lower cut around 250 GeV and has around ~ 50 particles per jet on average. When fine-tuning, the pre-trained model must adapt to these differences and learn to generate jets with fewer particles and the new kinematic cuts. The results for the p_T distributions and particle multiplicities show how the fine-tuned model smoothly interpolates between the AOJ and JetClass as a function of the training sample size D . Furthermore, the generation quality improves with increasing training sample size, achieving almost optimal results with a reduced sample size of 100 K jets for the p_T distribution and 10 K for the multiplicities. In contrast, the randomly initialized models trained from scratch require a significantly larger dataset, approximately an order of magnitude more jets, to achieve optimal performance. For the jet mass and the substructure observable τ_{21} , the fine-tuning performance saturates with a remarkably small number of training samples, resulting in an almost flat dependence of the metrics on D . Specifically, fine-tuning requires only about 1000 jets to reach optimal performance, whereas training from scratch requires significantly more data to achieve comparable results. This is not too surprising, as the transfer from AOJ to JetClass for these features does not require much effort, given the similarity in the shapes of these distributions.

Next, we examine the results in figure 4, which correspond to top-quark jet generation. Since the AOJ dataset is completely dominated by QCD jets, the transfer learning task between AOJ and JetClass becomes more challenging due to the change in jet type. For instance, fine-tuning must map the low-mass Sudakov

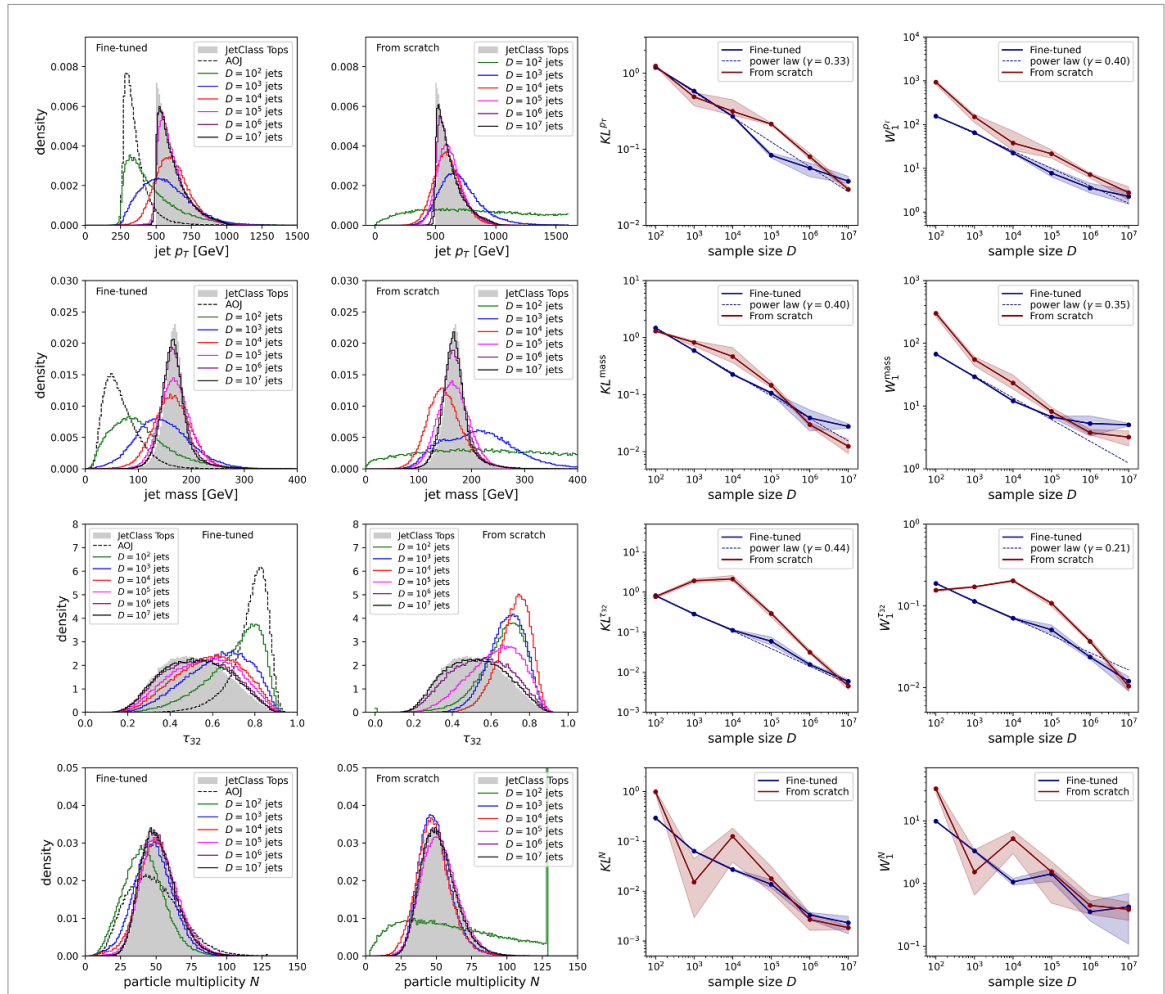


Figure 4. A comparison of the generation quality of models trained on top-quark jets from JetClass, for different training sample sizes D . The performance of a foundation model pre-trained on AJO (far left) is compared to a model trained from scratch (center left). The generation quality is compared across several high level features of the jets. Two quantitative metrics, the Kullback–Leibler divergence (center right) and Wasserstein-1 distance (far right), are computed as a function of the training sample size D to compare how well the generated jets from each model matches the target distribution from JetClass. We also report the mean value and the envelopes over 5 trainings with different random seeds. When applicable, we also show power-law fits $\propto D^{-\gamma}$ to the metrics.

peak characteristic of QCD jets to the top-quark resonance peak at $m_t = 175$ GeV. Additionally, the model must transition from generating 1-prong QCD jets to 3-pronged top-quark jets, as reflected in the horizontal gap between the peaks in the τ_{32} substructure distributions. Despite these obstacles, we find that transfer learning consistently outperforms training from scratch when using smaller datasets. For example, the top-quark mass, represented by the mode in the jet mass distribution, is learned with good precision by the fine-tuned model using a training dataset of only 10 K jets. In contrast, the model trained from scratch requires an order of magnitude more data to achieve comparable performance. A similar trend is observed for the τ_{32} distribution, where transfer learning morphs the AJO jets into JetClass top-quark jets, achieving competitive results with only 100 K training jets. In comparison, models trained from scratch fail to generate jets with a reliable substructure for training sample sizes below 1 M jets.

3.4. Power-law scaling

Finally, we observe that the performance metrics for the fine-tuned models, when not in the saturated regime, seem to follow a power-law scaling of the form $D^{-\gamma}$ with respect to the training sample size D . Neural scaling laws, such as these, have attracted broad interest and have been analyzed extensively in various settings in recent years, see [56] for a nice overview of the ML literature and original references. In particular, they have been observed in the context of jet classification [57]. We find that the exponents for a wide range of high-level observables are consistently within the range $0.2 < \gamma < 0.45$ (see dashed lines in figures 3 and 4). This scaling behavior holds for both the KL and Wasserstein-1 metrics and applies to both QCD and top-quark, suggesting a form of *universality*. The only exception to this scaling is the particle multiplicity distribution, which shows a more irregular dependence on the training sample size. Previous scaling law

behavior has been observed as a function of the loss of the model, not on such hand-crafted metrics. However, due to the different tokenizers used in the from-scratch and the AOJ fine-tuned model, the losses for the generation objective are not directly comparable, making statements of true scaling comparisons difficult. We leave further investigation of this apparent scaling behavior to future work.

In contrast, the randomly initialized models trained from scratch are not seen to follow such universal scaling across most physical observables. The reason behind this may be that these operate in the small-data regime [58], where their performance is limited by the lack of sufficient training data. In this regime, models can only perform marginally better than random guessing, as each additional sample provides limited new information for learning. The breaking of scaling is particularly evident for the jet substructure features, such as τ_{21} for QCD and τ_{32} for tops, and to a lesser extent for the jet mass, where the performance metrics exhibit irregular and dataset-dependent scaling as the training sample size increases. Only after the model has seen sufficient training data, approximately around 1 M jets, does the training stabilize, resulting in reliable jet substructure and jet mass modeling. While further studies into these scaling laws need to be performed, the observed discrepancy highlights the possible advantage of pre-training in achieving a more robust and systematic improvement in generative performance.

4. Conclusion

We are releasing the AOJs dataset, which comprises 178 million jets from CMS open data, in a standard machine learning-friendly format. This dataset provides an excellent resource for exploring foundation models and other machine learning techniques that use large, unlabeled datasets for pre-training.

In addition, we present the first jet-based foundation model in high-energy physics trained on real collider data from the AOJ dataset. By using this rich dataset to pre-train a foundation model, we show that fine-tuning for specific tasks—such as generating jets of different types in different kinematic regions—yields significant performance improvements over models trained from scratch, and boosts the achievable downstream performance on small labeled datasets. In particular, the generative performance of our foundation model scales predictably with training sample size, facilitating effective transfer learning. This demonstrates the power of foundation models to provide improved performance even over significant domain shifts.

We hope that this study will encourage further engagement with open data from LHC experiments, and promote advances that bridge phenomenological studies and deployment in actual experiments.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: <https://doi.org/10.25592/uhhfdm.16505>.

Acknowledgments

This work was initiated at the Aspen Center for Physics, supported by National Science Foundation Grant PHY-2210452. We would like to thank Alexander Mück for discussions. The research of MK and HR-G is supported by the Deutsche Forschungsgemeinschaft DFG under Grant 396021762 – TRR 257: Particle physics phenomenology after the Higgs discovery. JB, AH and GK are supported by the DFG under the German Excellence Initiative—EXC 2121 Quantum Universe – 390833306, and by PUNCH4NFDI—project number 460248186. DAF, IP, and DS are supported by DOE Grant DOE-SC0010008. OA is supported by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. LA is supported by the University of Bologna. Additionally, we acknowledge support from the Maxwell computational resources at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany, and computing resources provided by RWTH Aachen University under project rwth0934. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award HEP-ERCAP0027491.

Code

The code that was used to create ASPENOPENJETS from CMS open data can be found at <https://github.com/OzAmram/AOJProcessing>, and the code used for the OMNIJET- α model and its training can be found at https://github.com/uhh-pd-ml/omnijet_alpha.

Appendix. CMS samples

MC samples provided by CMS were used to estimate the composition of the AOJ sample. CMS uses a variety of generators for the simulation different physics processes, including MADGRAPH5_AMC@NLO [43], PYTHIA [44], and POWHEG [59–61]. Each sample uses PYTHIA with the underlying event tune CP5 [62] to simulate the parton shower and hadronization. All samples use the NNLO NNPDF 3.1 parton distribution functions [63–65]. Some physics processes are split into several samples covering different kinematic regions. Each sample is normalized to the appropriate cross section.

To estimate the number of fully merged W , Z , top, and $H \rightarrow b\bar{b}$ jets the corresponding MC samples were used. A matching criteria was applied based on truth-level information from the generator to ensure the estimate reflected the number of fully-merged jets of each type. For each heavy resonance, generator-level information was used to check that all the quarks from the heavy resonance decay were within $\Delta R < 0.8$ of the reconstructed AK8 jet found by our selection. The remaining 99.5% of jets not estimated to be heavy resonance decays are assumed to be QCD. The resulting numbers should be considered rough estimates, as the MC modeling of these signals is imperfect, and the estimates do not incorporate corrections to the simulated trigger and reconstruction efficiency typically employed in physics analyses by CMS.

Table 2 lists the CMS data and MC samples used and their corresponding cross sections.

Table 2. A list of the datasets used to construct AOJ and the Monte Carlo samples used to estimate its composition.

Process	CMS dataset name	Cross section (pb)
Data	/JetHT/Run2016G-UL2016_MiniAODv2-v2/MINIAOD	—
	/JetHT/Run2016H-UL2016_MiniAODv2-v2/MINIAOD	—
QCD	/QCD_Pt_300to470_TuneCP5_13TeV_pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v1/MINIAODSIM	7823
	/QCD_Pt_470to600_TuneCP5_13TeV_pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v1/MINIAODSIM	648.2
	/QCD_Pt_600to800_TuneCP5_13TeV_pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v1/MINIAODSIM	186.9
	/QCD_Pt_800to1000_TuneCP5_13TeV_pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v1/MINIAODSIM	32.3
$t\bar{t}$	/TTToHadronic_TuneCP5_13TeV-powheg-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v1/MINIAODSIM	378.5
W + jets	/TTToSemiLeptonic_TuneCP5_13TeV-powheg-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v1/MINIAODSIM	365.2
	/WJetsToQQ_HT-400to600_TuneCP5_13TeV-madgraphMLM-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v2/MINIAODSIM	315.6
	/WJetsToQQ_HT-600to800_TuneCP5_13TeV-madgraphMLM-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v2/MINIAODSIM	68.6
	/WJetsToQQ_HT-800toInf_TuneCP5_13TeV-madgraphMLM-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v2/MINIAODSIM	34.9
Z + jets	/ZJetsToQQ_HT-400to600_TuneCP5_13TeV-madgraphMLM-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v2/MINIAODSIM	145.4
	/ZJetsToQQ_HT-600to800_TuneCP5_13TeV-madgraphMLM-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v2/MINIAODSIM	34.0
$H \rightarrow b\bar{b}$	/ZJetsToQQ_HT-800toInf_TuneCP5_13TeV-madgraphMLM-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v2/MINIAODSIM	18.7
	/GluGluHToBB_Pt-200ToInf_M-125_TuneCP5_MINLO_13TeV-powheg-pythia8/RunIISummer20UL16MiniAODv2-106X_mcRun2_asymptotic_v17-v2/MINIAODSIM	0.27

References

- [1] Bommasani R et al 2021 (available at: <https://crfm.stanford.edu/assets/report.pdf>)
- [2] Brown T et al 2020 *Advances in Neural Information Processing Systems* vol 33 (Curran Associates, Inc.) pp 1877–901
- [3] Devlin J, Chang M-W, Lee K and Toutanova K 2019, arXiv:1810.04805 [cs.CL]
- [4] Touvron H et al 2023, arXiv:2302.13971 [cs.CL]
- [5] Saharia C et al (Curran Associates Inc.) 2022 arXiv:2205.11487
- [6] Radford A et al 2021 arXiv:2103.00020 [cs.CV]
- [7] Jia C, Yang Y, Xia Y, Chen Y, Parekh Z, Pham H, Le Q V, Sung Y, Li Z and Duerig T 2021, arXiv:2102.05918
- [8] Kishimoto T, Morinaga M, Saito M and Tanaka J 2023 *37th Conf. on Neural Information Processing Systems*
- [9] Qu H, Li C and Qian S 2022, arXiv:2202.03772 [hep-ph]
- [10] Heinrich L, Golling T, Kagan M, Klein S, Leigh M, Osadchy M and Raine J A 2024 arXiv:2401.13537 [hep-ph]
- [11] Birk J, Hallin A and Kasieczka G 2024 *Mach. Learn. Sci. Tech.* **5** 035031
- [12] Harris P, Kagan M, Krupa J, Maier B and Woodward N 2024 arXiv:2403.07066 [hep-ph]
- [13] Mikuni V and Nachman B 2024 arXiv:2404.16091 [hep-ph]
- [14] CMS Collaboration 2022 Multijet primary dataset in AOD format from RunA of 2011 (/MultiJet/Run2012A-22Jan2013-v1/AOD) (available at: <http://dx.doi.org/10.7483/OPENDATA.CMS.WING.7QKV>)
- [15] CMS Collaboration 2022 Multijet primary dataset in AOD format from RunA of 2012 (/MultiJet/Run2012A-22Jan2013-v1/AOD) (available at: <http://dx.doi.org/10.7483/OPENDATA.CMS.WING.7QKV>)
- [16] CMS Collaboration 2024 JetHT primary dataset in MINIAOD format from RunG of 2016 (/JetHT/Run2016G-UL2016_MiniAODv2-v2/MINIAOD) (available at: <http://dx.doi.org/10.7483/OPENDATA.CMS.1KTG.X0W4>)
- [17] CMS Collaboration 2024 JetHT primary dataset in MINIAOD format from RunH of 2016 (/JetHT/Run2016H-UL2016_MiniAODv2-v2/MINIAOD) (available at: <http://dx.doi.org/10.7483/OPENDATA.CMS.LT9E.T7RQ>)
- [18] ATLAS Collaboration 2024 DAOD_PHYSLITE format 2015-2016 Open Data for Research from the ATLAS experiment (available at: <http://dx.doi.org/10.7483/OPENDATA.ATLAS.9HK7.P5SI>)
- [19] Larkoski A, Marzani S, Thaler J, Tripathi A and Xue W 2017 *Phys. Rev. Lett.* **119** 132003
- [20] Tripathi A, Xue W, Larkoski A, Marzani S and Thaler J 2017 *Phys. Rev. D* **96** 074003
- [21] Komiske P T, Mastandrea R, Metodiev E M, Naik P and Thaler J 2020 *Phys. Rev. D* **101** 034009
- [22] Dolan M J, Gargalionis J and Ore A 2023 arXiv:2312.03434 [hep-ph]
- [23] Qu H, Li C and Qian S 2022 JetClass: a large-scale dataset for deep learning in jet physics (available at: <http://dx.doi.org/10.5281/zenodo.6619768>)
- [24] Chatrchyan S et al (CMS) 2008 *JINST* **3** S08004
- [25] CMS Collaboration 2014 Jet primary dataset in AOD format from RunB of 2010 (/Jet/Run2010B-Apr21Reco-v1/AOD) (available at: <http://dx.doi.org/10.7483/OPENDATA.CMS.3S7F.2E9W>)
- [26] Petrucciani G, Rizzi A and Vuosalo C (CMS) 2015 *J. Phys.: Conf. Ser.* **664** 7
- [27] Peruzzi M, Petrucciani G, Rizzi A and (the CMS Collaboration) 2020 *J. Phys.: Conf. Ser.* **1525** 012038
- [28] Sirunyan A M et al (CMS) 2020 *JINST* **15** 10017
- [29] Khachatryan V et al (CMS) 2017 *JINST* **12** 01020
- [30] Sirunyan A M et al (CMS) 2017 *JINST* **12** 10003
- [31] Cacciari M, Salam G P and Soyez G 2008 *J. High Energy Phys.* **JHEP04(2008)063**
- [32] Cacciari M, Salam G P and Soyez G 2012 *Eur. Phys. J. C* **72** 1896
- [33] Sirunyan A M et al (CMS) 2020 *JINST* **15** 09018
- [34] Bertolini D, Harris P, Low M and Tran N 2014 *J. High Energy Phys.* **JHEP10(2014)059**
- [35] Khachatryan V et al (CMS) 2017 *JINST* **12** 02014
- [36] CMS Collaboration 2024 PFNano producer tool for CMS 2016 (available at: <https://opendata.cern.ch/record/12504>)
- [37] CMS Collaboration, 2024 Monte Carlo simulations of 2016 data (available at: <https://opendata.cern.ch/search?q=&f=experiment%3ACMS&f=type%3ADataset%2Bsubtype%3ASimulated&f=year%3A2016>)
- [38] Larkoski A J, Marzani S, Soyez G and Thaler J 2014 *J. High Energy Phys.* **JHEP05(2014)146**
- [39] Navas S et al (Particle Data Group) 2024 *Phys. Rev. D* **110** 030001
- [40] Thaler J and Van Tilburg K 2011 *J. High Energy Phys.* **JHEP03(2011)015**
- [41] Qu H and Gouskos L 2020 *Phys. Rev. D* **101** 056019
- [42] CMS 2020 (available at: <https://cds.cern.ch/record/2707946>)
- [43] Allwall J, Frederix R, Frixione S, Hirschi V, Maltoni F, Mattelaer O, Shao H S, Stelzer T, Torrielli P and Zaro M 2014 *J. High Energy Phys.* **JHEP07(2014)079**
- [44] Sjöstrand T, Ask S, Christiansen J R, Corke R, Desai N, Ilten P, Mrenna S, Prestel S, Rasmussen C O and Skands P Z 2015 *Comput. Phys. Commun.* **191** 159
- [45] de Favereau J, Delaere C, Demin P, Giammanco A, Lemaître V, Mertens A and Selvaggi M (DELPHES 3) 2014 *J. High Energy Phys.* **JHEP02(2014)057**
- [46] Radford A, Narasimhan K, Salimans T and Sutskever I 2018 (available at: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)
- [47] van den Oord A, Vinyals O and Kavukcuoglu K 2018 arXiv:1711.00937 [cs.LG]
- [48] Bao H, Dong L, Piao S and Wei F 2022 arXiv:2106.08254 [cs.CV]
- [49] Huh M, Cheung B, Agrawal P and Isola P 2023 arXiv:2305.08842 [cs.LG]
- [50] Wright L 2019 GitHub repository
- [51] Virtanen P et al 2020 *Nat. Methods* **17** 261
- [52] Krause C and Shih D 2023 *Phys. Rev. D* **107** 113003
- [53] Kansal R, Li A, Duarte J, Chernyavskaya N, Pierini M, Orzari B and Tomei T 2023 *Phys. Rev. D* **107** 076017
- [54] Das R, Favaro L, Heimerl T, Krause C, Plehn T and Shih D 2024 *SciPost Phys.* **16** 031
- [55] Hernandez D, Kaplan J, Henighan T and McCandlish S 2021 arXiv:2102.01293
- [56] Villalobos P 2023 (available at: <https://epoch.ai/blog/scaling-laws-literature-review>) (Accessed: 13 December 2024)
- [57] Batson J and Kahn Y 2023 arXiv:2312.02264 [hep-ph]
- [58] Hestness J, Narang S, Ardalani N, Diamos G, Jun H, Kianinejad H, Patwary M M A, Yang Y and Zhou Y 2017 arXiv:1712.00409

- [59] Nason P 2004 *J. High Energy Phys.* [JHEP 0411:040](#)
- [60] Frixione S, Nason P and Ridolfi G 2007 *J. High Energy Phys.* [JHEP09\(2007\)126](#)
- [61] Alioli S, Nason P, Oleari C and Re E 2010 *J. High Energy Phys.* [JHEP06\(2010\)043](#)
- [62] Sirunyan A M *et al* (CMS) 2020 *Eur. Phys. J. C* [80 4](#)
- [63] Ball R D, Del Debbio L, Forte S, Guffanti A, Latorre J I, Rojo J and Ubiali M 2010 *Nucl. Phys. B* [838 136](#)
- [64] Ball R D *et al* (NNPDF) 2015 *J. High Energy Phys.* [JHEP04\(2015\)040](#)
- [65] Ball R D *et al* (NNPDF) 2017 *Eur. Phys. J. C* [77 663](#)



PAPER

OPEN ACCESS

RECEIVED
6 February 2026REVISED
21 April 2026ACCEPTED FOR PUBLICATION
20 May 2026PUBLISHED
4 June 2026

Original content from
this work may be used
under the terms of the
[Creative Commons
Attribution 4.0 licence](#).

Any further distribution
of this work must
maintain attribution to
the author(s) and the title
of the work, journal
citation and DOI.



Enhancing next token prediction based pre-training for jet foundation models

Joschka Birk^{1,*} , Anna Hallin¹ , Gregor Kasieczka¹ , Nikol Madzharova¹ , Ian Pang²
and David Shih²

¹ Institut für Experimentalphysik, Universität Hamburg, 22761 Hamburg, Germany

² NHETC, Department of Physics and Astronomy, Rutgers University, Piscataway, NJ 08854, United States of America

* Author to whom any correspondence should be addressed.

E-mail: joschka.birk@uni-hamburg.de, anna.hallin@uni-hamburg.de, gregor.kasieczka@uni-hamburg.de,
nikol.madzharova@studium.uni-hamburg.de, ian.pang@physics.rutgers.edu and shih@physics.rutgers.edu

Keywords: particle physics, foundation models, pre-training, generation, classification, next token prediction, masked modeling

Abstract

Next token prediction (NTP) is an attractive pre-training task for jet foundation models, in that it is simulation free and enables excellent generative capabilities that can transfer across datasets. Here we study multiple improvements to NTP, building on the initial work of OmniJet- α . Instead of tokenizing particles and subsequently only using the token-IDs as the model input for both the generative and the classification task, we adopt a hybrid setup, which allows us to use continuous feature vectors as model input while only using token-IDs in the NTP target. Secondly, we explore a combined pre-training strategy that combines masked particle modeling and generative learning objectives. Taken together, these changes greatly improve the performance in downstream classification tasks without any loss in generative performance.

1. Introduction

Foundation models [1] applied to high energy physics (HEP) data have recently received substantial interest, motivated by their ability to generalize to previously unseen datasets and tasks. Jet physics in particular has been a key target for the development of HEP foundation models (see [2] for a recent review), due to its wide practical relevance across LHC and future collider experiments, and its well-established role as benchmark for machine learning algorithms [3].

Much of the activity so far in the development of foundation models for jets has centered around exploring different approaches to pre-training and fine-tuning in order to optimize the expressivity of the learned representations and the performance on downstream tasks. For example, there have been many different proposals for the pre-training objective. These range from predicting a missing entry, based either on masking [4–8] or next token prediction (NTP) [9], to classification [10–14], data augmentation and contrastive learning [15, 16] or a combination of tasks [11–13]. Despite all these efforts, it remains unclear to what extent different pre-training methods produce representations that align well with the demands of downstream tasks. Different pre-training objectives correlate strongly with performance on downstream tasks: for instance some approaches enable classification but not generation or vice versa. This is clearly an unsatisfactory situation for the HEP foundation model paradigm.

Another dividing line is the reliance on ground-truth information in the training phase, which determines whether they can in-principle be trained using data or whether simulation is required. While simulation-based approaches [10–15, 17] can leverage supervised training on high-fidelity simulations, simulation-free approaches [4–9, 16, 18] have the potential to be trained directly on the copious amounts of real LHC data.

This work focuses on OmniJet- α [9] as the only model with demonstrated generative capabilities and transfer learning between tasks that can be fully trained on data. However, these advantages so far came at a price, namely a sub-optimal performance when considering downstream classification tasks. Here, we identify and improve upon two limitations of the original OmniJet- α approach, stemming from it

being an autoregressive GPT-like model. In both cases, we modify the architecture to align more closely with the nature of our physics data, while maintaining the attractive simulation-free and generation-capable properties.

First, we show that using tokenized input during the classifier training harms classification ability. The main reason for tokenization is that it improves generative pre-training and generative downstream performance. However, due to information loss and possible artifacts [19], tokenization might not be the most optimal choice. To circumvent these issues, we present an improved hybrid setup for NTP based pre-training, where the input feature vectors are continuous while the pre-training targets remain token-IDs. Allowing for continuous input significantly increases the classification capabilities of our model, while keeping the token-IDs as targets for the pre-training preserves the generative performance.

Second, we revisit the pre-training task. It is known from natural language processing (NLP) that models based on causal attention and NTP excel at generation, whereas bi-directional approaches using masking produce rich contextual embeddings. Recent works that have focused on combining the two training objectives include [20–26]. While OmniJet- α has strong generative capabilities due to its NTP pre-training, we hypothesize that tasks such as jet classification require a contextual understanding of the jet constituents which may be better served by bi-directional pre-training like masked particle modeling (MPM) [4, 5] and other methods based on masked prediction [6–8]. Inspired by the results from the NLP literature, we investigate the performance of both NTP and MPM during pre-training and subsequent fine-tuning to classification, including a setup that combines both training objectives. We show how pre-training on both next and masked token prediction simultaneously leads to a drastic improvement of downstream classification performance, while maintaining the generative capabilities.

The rest of this paper adheres to the following structure: section 2 describes the used datasets, reviews the original OmniJet- α method and its limitations, and details the modifications we introduce to address them. Section 3 reports the performance gains from (i) adapting to a continuous model input and (ii) updating the pre-training objective to combine next token and masked token prediction. Our conclusions are presented in section 4. Additional details, analyses and ablation studies are provided in appendices A–D.

2. Setup

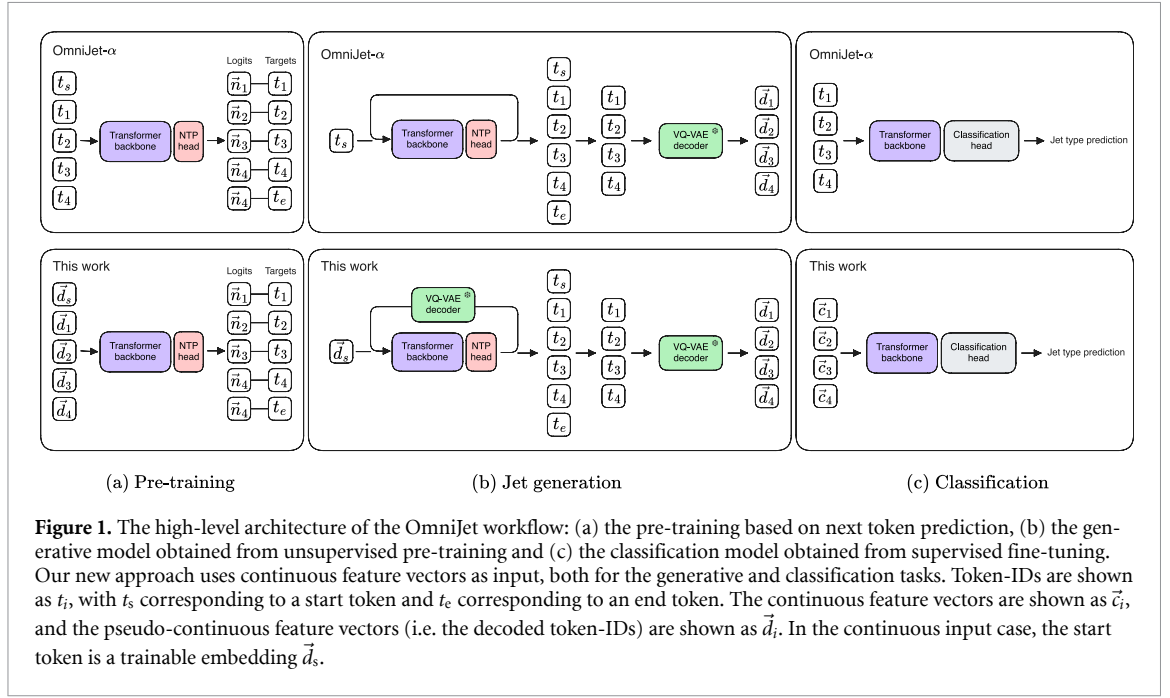
The model is implemented in PYTORCH [27] and LIGHTNING [28]. Furthermore, we use the open source software libraries in [29–40]. The overall architecture of the model is similar to the one used in OmniJet- α , with a few minor updates outlined in appendix A.

2.1. Datasets

Two datasets are used in our studies: the JetClass [10, 41] dataset and the top tagging dataset [3, 42].

The JetClass dataset contains 100 M training jets from simulated proton-proton collisions at the LHC. It contains ten different jet types, with the individual jet types corresponding to one background jet type (quark- or gluon-initiated jets) and nine signal jet types where the jet originates from the decay of a top quark, a W , Z or Higgs boson. The individual jet types are: q/g , $t \rightarrow bqq'$, $t \rightarrow b\ell\nu$, $W \rightarrow qq'$, $Z \rightarrow q\bar{q}$, $H \rightarrow b\bar{b}$, $H \rightarrow c\bar{c}$, $H \rightarrow gg$, $H \rightarrow 4q$, $H \rightarrow \ell\nu qq'$. The events are simulated with MadGraph5_aMC@NLO [43]. Pythia8 [44, 45] is used for parton showering and hadronization, and a simplified detector simulation is performed with Delphes [46] using the CMS [47] detector card. The particles are clustered with the anti- k_T [48] algorithm with a distance parameter of $R = 0.8$, and only jets with a transverse momentum in the range of $500 \text{ GeV} < p_T < 1000 \text{ GeV}$ and a pseudorapidity of $|\eta| < 2.0$ are included in the dataset. The JetClass dataset is a widely used, public benchmark dataset for studies in the domain of pre-training methods for jet physics. The p_T range used in JetClass corresponds to boosted jets, where heavy particles (t , W , Z , H) are fully contained within the jet. Further details on the JetClass dataset can be found in [10].

The top tagging dataset contains q/g jets as well as $t \rightarrow bqq'$ jets. This dataset is significantly smaller than the JetClass dataset, with 1.2 M training jets. It contains jets extracted from LHC events that were simulated with Pythia8 [44, 45] and Delphes [46] using the ATLAS [49] detector card. The jets are clustered with the anti- k_T [48] algorithm with a distance parameter of $R = 0.8$, and are required to have $|\eta| < 2.0$ and a transverse momentum in the range of $550 \text{ GeV} < p_T < 650 \text{ GeV}$. While the p_T -ranges of the two datasets used here overlap, earlier work such as [18] showed successful transfer learning across datasets with different p_T ranges.



2.2. Original OmniJet- α method

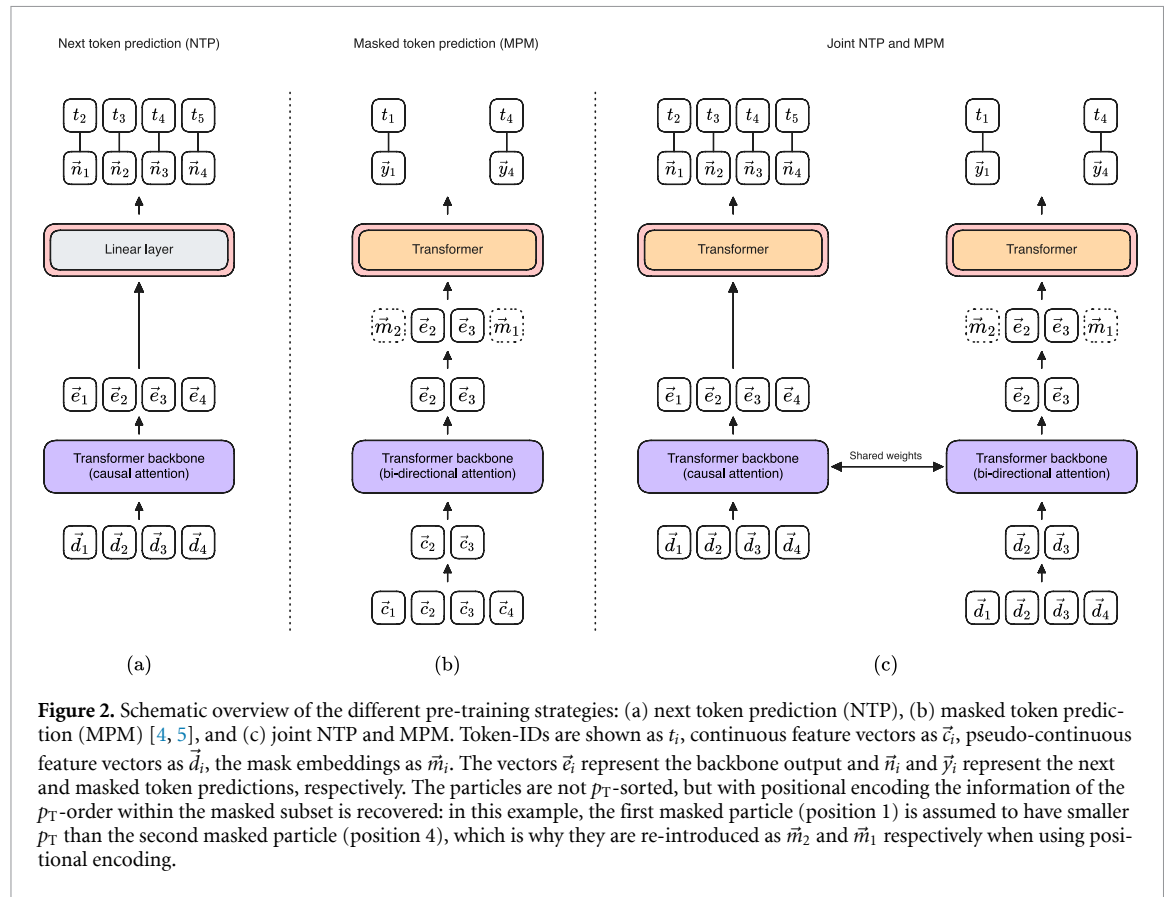
An overview of the method of the original work is shown in figure 1 (upper row). In the original OmniJet- α work jet constituents are first tokenized with a vector quantized autoencoder (VQ-VAE) [4, 50, 51] to obtain a discrete representation of the continuous input features, in the following referred to as *token-IDs*, denoted as t_i . A transformer architecture is used for both the encoder and decoder of the VQ-VAE, which means that both the encoding and the decoding of a particle is conditioned on the other particles present in the sequence. Thus, each particle is mapped to a single token-ID, but the other particles present in the sequence can influence the tokenization process. This allows the tokenizer to cover the physical space more accurately as opposed to unconditional tokenization. The VQ-VAE is trained as an individual step before any pre-training is performed, and is frozen during the pre-trainings. The pre-training is performed on the NTP task. Afterwards, the integer number token-IDs are used as input for both the generative and classification tasks.

A key advantage of the original OmniJet- α setup is that generative pre-training produces good generative models that can be transferred to different datasets with minimal fine-tuning [18], and earlier studies have shown that such models are directly useful for anomaly detection [11, 12, 52]. Beyond this immediate utility, the same pre-training was shown to significantly improve classification performance relative to training from scratch. However, the absolute classification performance obtained in this way still falls short of the current state-of-the-art. As we will show with the modifications to OmniJet- α detailed in the next subsection, this performance gap arises from the original design choice to use the discrete token-IDs as model input, as well as from the limited contextual representation learned under a NTP objective.

2.3. Modifications

2.3.1. Continuous-input NTP

The main advantage of using continuous input vectors is that the downstream classification task can operate directly on the full resolution feature vectors $\vec{c}_i$, ensuring that the model does not suffer from tokenization-induced loss of information in this task. With this goal of improving downstream classification performance, we use continuous feature vectors as model inputs while token-IDs serve only as NTP targets, following the hybrid setup employed in prior works. The general framework is shown in figure 1(bottom row): during pre-training, the model acts on pseudo-continuous feature vectors, denoted in figure 1 as $\vec{d}_i$, which refers to the continuous representations of the tokenized inputs (i.e. the decoded token-IDs). During autoregressive jet generation, the decoding of token-IDs is integrated into the autoregressive generation loop. Note that this additional decoding step is not needed during training, as the whole sequence of particle tokens is fed to the model at once.



2.3.2. Masked token prediction

Following masked modeling approaches in NLP and computer vision [53–58], masked modeling was introduced for jet physics in the form of MPM [4, 5]. Although MPM does not produce a generative model out of the box, it has demonstrated competitive classification performance. Since the pre-training strategies of MPM and our method differ mainly in whether the model predicts masked particles or the next particle, it is a close relative to our method and we include it in our setup in order to study the effects of masked token versus NTP on the classification task performance.

In the MPM setup, shown in figure 2(b), a fraction of the input particles is masked. The masked particles are removed from the input and only the particles that survived this masking step are passed to the transformer backbone. After the backbone, the masked particles are reinserted into the sequence in the form of mask embeddings $\vec{m}_i$, which are trainable embeddings that are initialized with Gaussian noise and are learned during training. Positional encoding is applied to these reinserted mask embeddings via a relation between the index i and position in the p_T -sorting: the index i refers to the position of a masked particle in the p_T -sorted subset of masked particles. That is, the i th highest p_T masked particle is reintroduced to the jet (after the backbone) as $\vec{m}_i$. The model head is then tasked to predict the token-ID of the masked positions. The main difference in our implementation compared to the original MPM works is that we use a significantly smaller backbone model, which is described in more detail in appendix A.

A natural question is whether the benefits of MPM stem from its objective (predicting masked tokens) or some difference in the neural network architecture. The main difference is the use of bi-directional attention in MPM (required by the permutation symmetry that comes with treating the jets as point clouds), compared with the causal attention used in NTP (required by treating the jets as sequences). To address this, we implement a version of MPM that uses a causal attention mask in the backbone transformer as well as in the model head, in the following referred to as *MPM-Causal*. This setup preserves the causal attention structure of NTP while changing only the prediction objective, allowing for a controlled comparison between the two factors. A masking rate of 40% is used in all MPM(-Causal) trainings as this was found to be the optimal masking rate in [5].

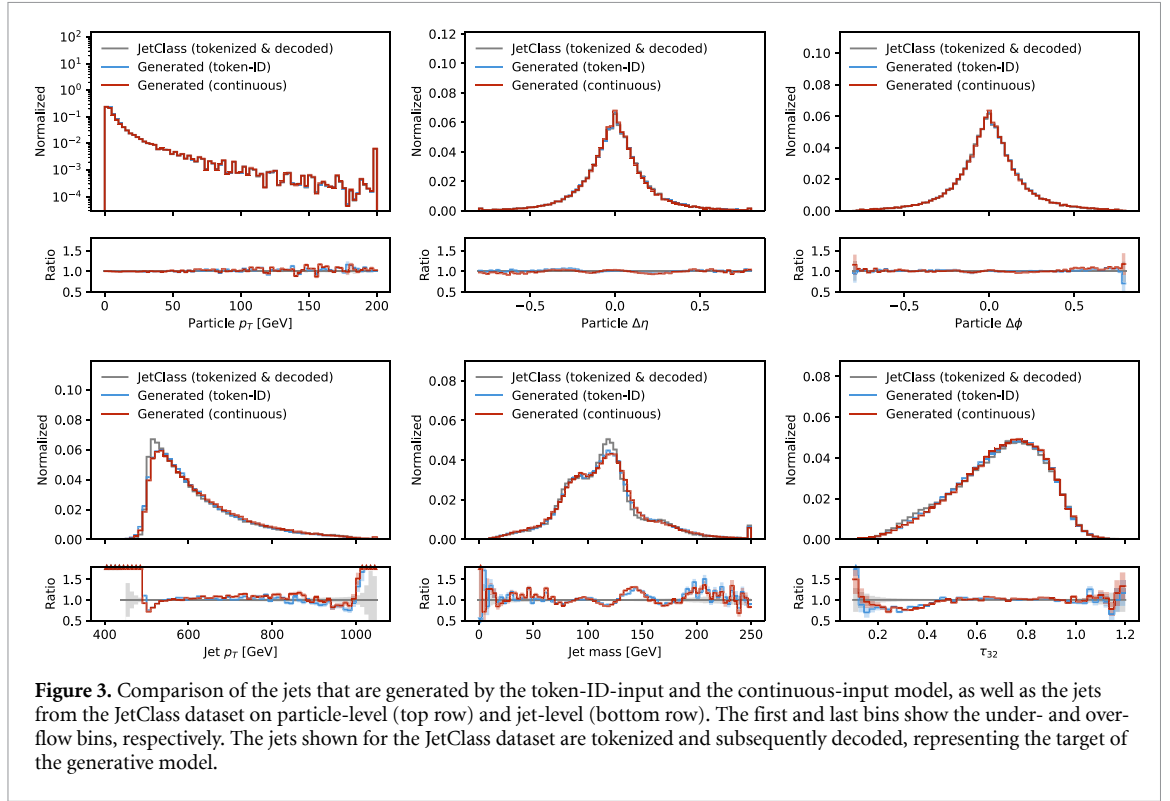


Figure 3. Comparison of the jets that are generated by the token-ID-input and the continuous-input model, as well as the jets from the JetClass dataset on particle-level (top row) and jet-level (bottom row). The first and last bins show the under- and over-flow bins, respectively. The jets shown for the JetClass dataset are tokenized and subsequently decoded, representing the target of the generative model.

2.3.3. Joint next and masked token prediction

To investigate if the advantages of both next and masked token prediction can be combined, we implement a joint pre-training. An illustration of the joint setup is shown in figure 2(c). This joint pre-training is realized by having two model heads, predicting the next token logits and the masked token logits, respectively. The two individual loss terms are simply added without any further modification. Two backbone forward passes are performed during pre-training. The forward pass for the masked token prediction is performed the same way as for pure MPM, i.e. with bi-directional attention and with positional encoding. For the NTP all jet constituents are used as input and the causal mask is applied. Furthermore, so as to not force the NTP backbone to align its output too strongly with the generative task, we use a transformer in the NTP head in addition to the simple unembedding layer that the original OmniJet- α was trained with. This allows the model to further modify the backbone representation with causal attention before projecting the representation to next token logits.

3. Results

We pre-train the backbone on the whole 100 M-jet training set of the JetClass dataset for each pre-training strategy. Afterwards, we evaluate the quality of the generative model (in cases where the pre-training involves NTP) and quantify the effectiveness of the different pre-training strategies by fine-tuning the pre-trained models on jet classification. Unless otherwise specified we restrict the set of input features to the kinematic features (particle p_T , $\Delta\phi$ and $\Delta\eta$), as the other features included in the JetClass dataset are not available in the top tagging dataset³.

3.1. Token-ID input vs continuous input

We first verify that the change of input structure from token-IDs to continuous feature vectors can be accomplished without sacrificing the performance of the corresponding generative model. The jet and particle-level distributions obtained from generated jets of the two different input types is compared to their target distributions in figure 3. For both models the particle-level distributions of the generated jets show very good agreement with the jets from the JetClass dataset and small deviations from the target

³ We investigate the effect of extending the feature set to include the particle-ID and trajectory displacement information of the jet constituents for a model pre-trained with NTP using continuous inputs, and observe, as expected, a significant improvement in the classification capabilities. This study is shown in appendix C.

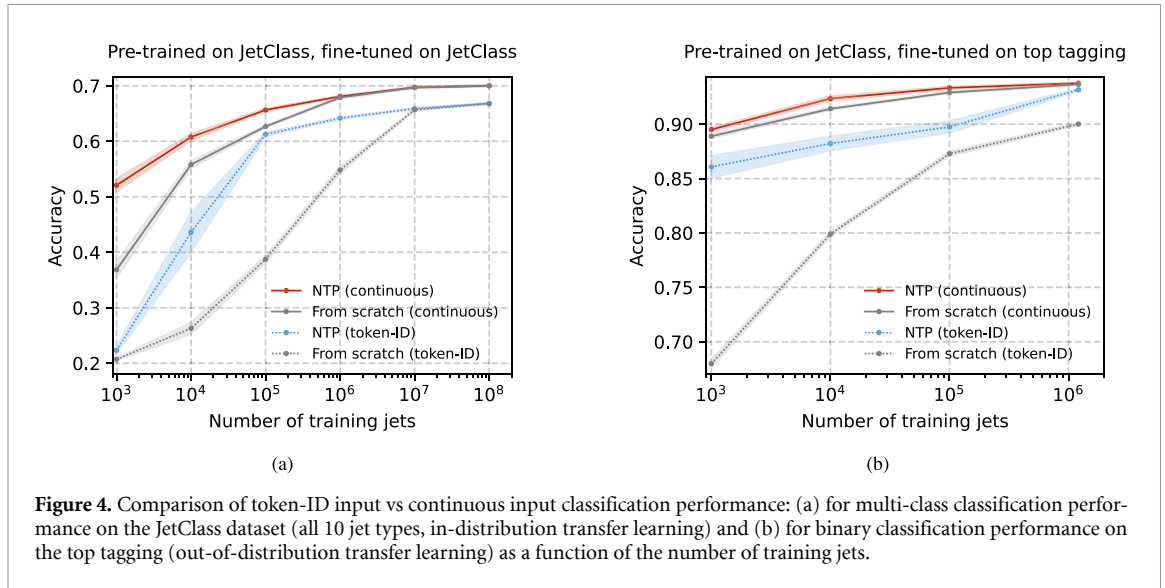


Figure 4. Comparison of token-ID input vs continuous input classification performance: (a) for multi-class classification performance on the JetClass dataset (all 10 jet types, in-distribution transfer learning) and (b) for binary classification performance on the top tagging (out-of-distribution transfer learning) as a function of the number of training jets.

distribution on jet-level features. The change in input structure (continuous feature vectors instead of token-IDs) thus does not harm the performance of the generative model.

The positive effect of the continuous input on the classification performance is shown in figure 4(a) for in-distribution and in figure 4(b) for out-of-distribution transfer learning. For in-distribution transfer learning, we fine-tune the model on multi-class classification (10 classes) on the JetClass dataset with training dataset sizes between 1 k and 100 M jets. As expected, the *from scratch* model, i.e. the baseline model without any pre-training, performs significantly better with continuous input features than with token-ID inputs for all dataset sizes. This demonstrates that the tokenization alone can significantly harm the classification performance.

The pre-trained models improve the performance compared to their respective from-scratch models in both the token-ID and the continuous case. In the token-ID input case, the pre-trained model outperforms the from-scratch model for all dataset sizes except for the largest dataset size. In the largest dataset size regime, at 100 M jets, the from-scratch model catches up with the fine-tuned model, which indicates that the model has seen enough training data to learn the relevant features from scratch and using the pre-trained weights as initialization does not provide a benefit anymore. Meanwhile, in the continuous input case, we still observe a performance gain from generative pre-training for small dataset sizes, while the from-scratch and fine-tuned model converge to a similar performance for training dataset sizes of 1 M jets or more.

For the out-of-distribution transfer learning we fine-tune the model on the binary classification task on the top tagging dataset. The dataset range for this study is between 1k and 1.2 M jets, with the latter being the full training dataset size of this dataset. We see very similar trends as for in-distribution transfer learning, with a clear performance gain when using continuous input features instead of token-ID input features. In this case, generative pre-training offers an improvement over the from-scratch baseline for the whole dataset range, both for the token-ID input and the continuous input case. While the continuous input classifier presented here is still outperformed by jet taggers that pre-train in a supervised manner directly on a classification task, like ParT [10], L-GATr [59, 60] or OmniLearned [13], this is expected as these studies, in contrast to state of the art models, do not make use of either physics inspired interaction features, class labels, or Lorentz-equivariant model architectures.

3.2. Comparison of different pre-training objectives

For the comparison of different pre-training strategies (NTP, MPM and joint), we pre-train with continuous inputs on the JetClass dataset with the kinematic-only feature set and fine-tune on the top tagging dataset. The results of this comparison are shown in figure 5. All investigated pre-training strategies lead to an improvement over the baseline for the whole range of training dataset sizes, except for the R_{30} metric⁴ at a training dataset size of 1000 jets where the fine-tuned NTP model performs slightly worse than the from-scratch baseline. This could be due to the fact that the difference between the pre-training task (NTP) and the target task (classification) is too large, with the model not being able to adjust to

⁴ The R_{30} metric is the background rejection at 30 % signal efficiency.

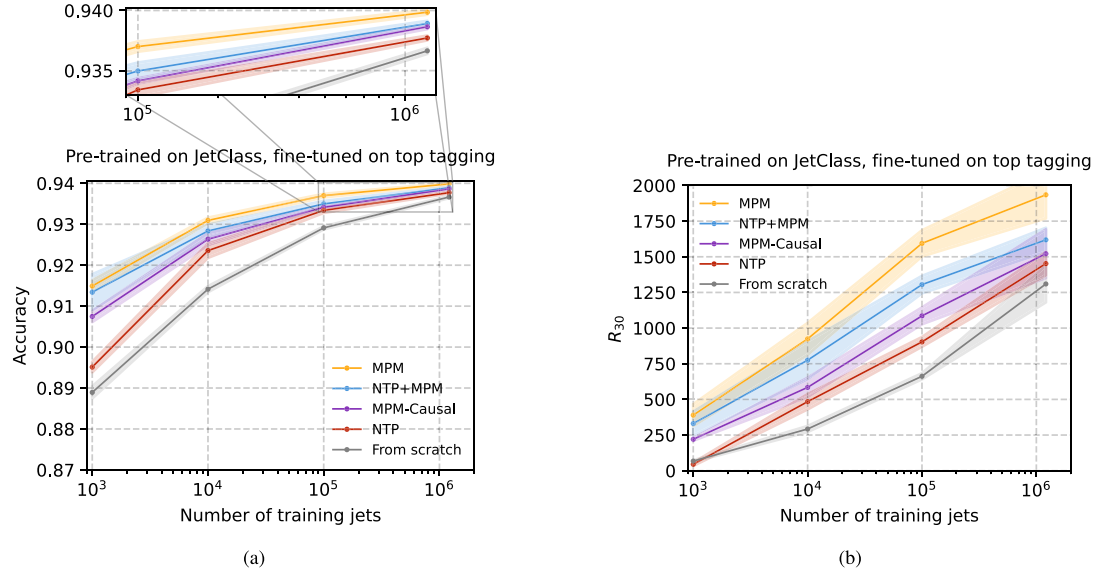


Figure 5. Classification performance on the top tagging dataset with different pre-training strategies: (a) the classifier accuracy and (b) the background (q/g -jet) rejection at 30 % signal (top-jet) efficiency. All models use continuous feature vectors as input.

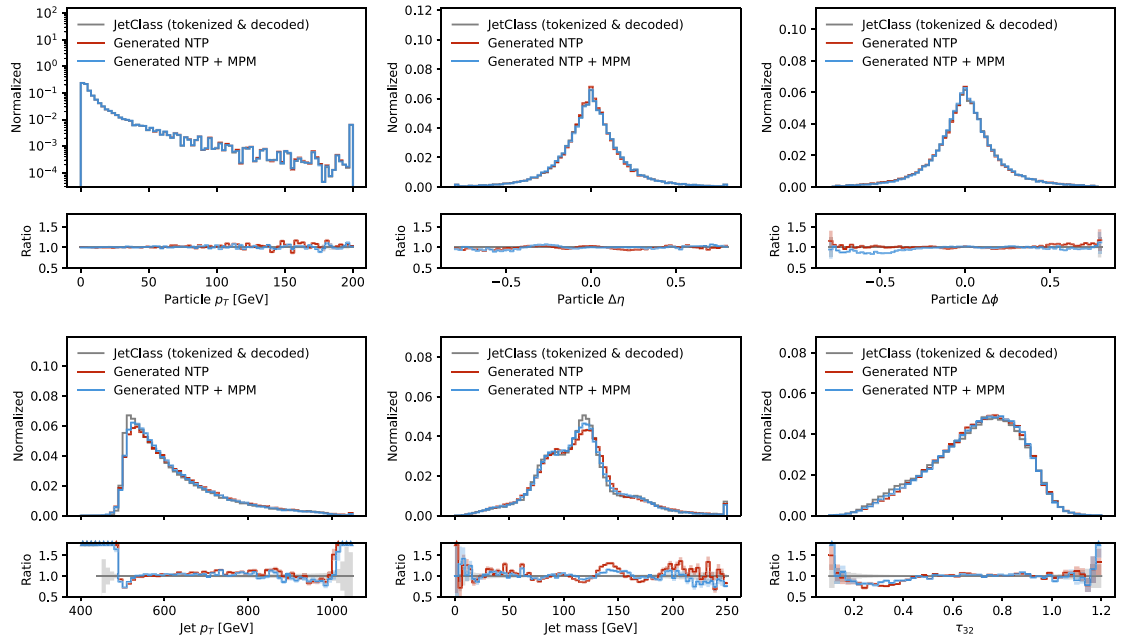


Figure 6. Comparison of the generative models trained with NTP only vs joint NTP and MPM. The jets shown for the JetClass dataset are tokenized and subsequently decoded, representing the target of the generative model.

the target task with only 1000 training jets during fine-tuning. Additional evidence for this interpretation is provided in appendix D, where fine-tuning with a fixed backbone shows that NTP-pretrained representations are substantially less aligned with classification than those obtained from MPM or joint pre-training.

While pre-training with NTP leads to a significant improvement over the from-scratch baseline, it is outperformed by the other pre-training strategies with the best performance being achieved by MPM. The MPM-Causal setup (i.e. predicting masked tokens with causal attention) performs similarly to the NTP setup. This indicates that the classification performance gains seen in MPM are not solely due to the masked prediction objective itself. The different attention mechanisms (bidirectional vs causal) also plays an important role. Apparently, bi-directional attention allows the model to develop a contextual understanding in a better way than causal attention does.

The joint pre-training clearly outperforms the standard NTP pre-training and results in a classification performance close to that of pure MPM. At the same time, as shown in figure 6, we observe that the generative capabilities of our model are not reduced by the joint training on NTP and MPM, with both models showing very similar agreement with the target distribution, both on particle level and on jet level.

4. Conclusion

When transplanting the methods of transformer-based foundation models from their origins in the modeling of language to their new home in solving problems in jet physics, their assumptions need to be revisited. For example, while the concept of tokens naturally fits data that can be seen as composed from discrete elements—such as words or word fragments—another representation might suit the typically continuous, high resolution read-outs of modern particle detectors better. Moreover, it is not yet clear which pre-training strategy or strategies best fit the specific downstream tasks that are of interest to particle physicists.

In this work, we have investigated these two questions: the effect of tokenizing inputs, and of setting useful pre-training tasks. We first found that a hybrid approach with respect to tokenization can combine the best of both worlds. While the learning target for the generative pre-training step remains tokenized, it uses continuous feature vectors with limited resolution (obtained by the decoder of the VQ-VAE that created the tokens in the first place) as input. This allows the model to keep its strong generative performance, while not restricting the input format to be tokenized. The model can then be fine-tuned directly to classify using full-resolution continuous inputs, leading to drastic improvement of classification performance as it is no longer limited to using token-IDs as input.

Next, we turned to the fairly subtle difference between predicting subsequent tokens (NTP) and predicting randomly masked-out tokens (MPM). Naively, the difference seems marginal. However, empirically, a non-trivial difference in performance can be observed. To better understand this behavior, we introduced the causally-masked version of MPM, allowing particles to only attend to their predecessors and thereby blocking the bi-directional attention. As this causal MPM version performs almost on-par with the generative NTP approach, one can conclude that the bi-directional attention of MPM plays a major role in aligning the learned representation for the classification task. Finally, we investigated a joint NTP/MPM pre-training scheme and showed that it is both possible and beneficial: combining both objectives improves on the downstream classification performance while maintaining the generative fidelity of pure NTP training. Evidently, enhancing the learned representation via some injection of MPM training does not adversely affect the generative power of NTP pretraining.

As both generative and predictive tasks are crucial applications of a successful foundation model of jet physics, the hybridization options developed in this work will aid the successful implementation of such a model. Beyond the ideas considered here, one might further investigate whether other tokenization schemes could yield performances surpassing the VQ-VAE—especially as the number of features is further increased. Alternatively, one could explore whether generation could even be achieved while fully avoiding discrete tokens [61]. Finally, these hybrid ideas could also point towards approaches that include physical priors more explicitly, while retaining the flexibility of foundation models.

Acknowledgments

The authors thank Oz Amram, Darius Farouhy, Michael Krämer, Alexander Mück, and Humberto Reyes-Gonzalez for stimulating exchanges regarding foundation models. We would also like to thank Thorsten Buss for valuable discussions throughout this project. JB, AH, GK, and NM are supported by the DFG under the German Excellence Initiative – EXC 2121 Quantum Universe - 390833306, and by PUNCH4NFDI—project number 460248186. IP and DS are supported by DOE grant DOE-SC0010008. We also acknowledge support via the SciFM consortium (05D25GU4) funded by the German Federal Ministry of Research, Technology, and Space (BMFTR) in the ErUM-Data action plan as well as the Hamburg VISTA/VISOR—Virtual Initiative for Science & Technology in AI—network. This research was supported in part by grant NSF PHY-2309135 to the Kavli Institute for Theoretical Physics (KITP).

For this work the HPC-cluster Hummel-2 at University of Hamburg was used. The cluster was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 498394658. Additionally, we acknowledge support from the Maxwell computational resources at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: <https://zenodo.org/records/6619768> [41], <https://zenodo.org/records/2603256> [42].

Code

The code for this paper is available at: github.com/uhh-pd-ml/enhancing-ntp4jets [62].

Appendix A. Model architecture and training details

A.1. Backbone architecture changes

Compared to OmniJet- α , which uses post-norm transformer blocks based on [63], we use pre-norm transformer [64] blocks in the backbone architecture as this choice is more common in modern transformer architectures. Furthermore, we add LAYERSCALE [65] to the architecture, but instead of initializing the per-channel weights with a small value as commonly done, we use a value of 1 as we found this initialization strategy to be more effective in practice in our trainings. We also add registers [66] to the backbone architecture which was found to slightly improve classification performance in initial studies. Overall we use a smaller backbone compared to OmniJet- α by using 8 transformer blocks with an embedding dimension of 128 instead of 3 transformer blocks with an embedding dimension of 256. Moreover, this leads to a much smaller backbone compared to the MPM work, which uses 8 transformer blocks with an embedding dimension of 512. The smaller model size in this work was chosen as we found that a larger backbone did not lead to better results in initial experiments. The token-ID-input backbone has 2.6 M parameters as opposed to 4.5 M parameters in the original OmniJet- α work. The number of parameters is further reduced in the continuous-input backbone, which has 1.6 M parameters. The much larger number of parameters in the token-ID-input case is due to the trainable token embedding layer for the token-ID input (which maps 8192 token-IDs to the embedding dimension 128 of the backbone), resulting in around 1 M additional parameters. Finally, the backbone model is conditioned on the number of particles, as the continuous-input generative model was found to struggle with correct end token prediction when not using this conditioning in early tests. For the case where the model acts as a generative model, we fit a kernel density estimate (KDE) on the distribution of the number of particles per jet, from which we sample to generate the conditional information for jet generation⁵. We combine jet-level information (i.e. the number of particles) with the per-particle information by first projecting both the jet-level and the particle-level feature vectors with a linear layer to the embedding dimension of the backbone, and then adding the jet-level embedding to each particle embedding. The details of the backbone architecture and hyperparameter choices are listed in table 1. Due to computational constraints, no dedicated hyperparameter optimization was performed. Instead, a few configurations were tested, and we selected configurations that were found to lead to stable training performance while also not requiring an unnecessarily large model.

A.2. Token prediction head

As in the original OmniJet- α work, we use a linear layer to project the backbone output to the logits of the token-IDs when training on the NTP target only. The token prediction head used for MPM, as well as the token prediction head used for the NTP prediction in the joint pre-training consists of two transformer blocks with the same embedding dimension as the backbone, followed by a linear layer to obtain the masked and next token logits, respectively.

A.3. Classification head

Compared to OmniJet- α we adapt the classification head to feature two class-attention blocks [10, 65] followed by a linear layer (instead of two linear layers with a summation over all particles between them) to obtain a more capable classification head. Furthermore, we do not apply the causal mask in the backbone in classification mode as this is an unnecessary restriction on the model's capacity when trained for classification. The classification head adds another 400 k parameters to the model, leading to a total of 2 M trainable parameters in the continuous-input classifier, which is similar in size to the OmniLearn [11, 12] or ParT [10] models.

⁵ The generated values from the KDE are clipped to be larger than zero and subsequently rounded to the closest integer number.

Table 1. Backbone model hyperparameters.

Hyperparameter	Value
Number of transformer blocks	8
Embedding dimension	128
LAYERSCALE initialization value	1.0
Optimizer	Ranger [68–70]
Learning rate	0.001
Batch size (if $N_{\text{train}} > 10\,000$)	1000
Batch size (if $N_{\text{train}} \leq 10\,000$)	100

Table 2. VQ-VAE model hyperparameters. For further information on α , β and ν see [37, 51].

Hyperparameter	Value
Number of transformer blocks (encoder)	4
Number of transformer blocks (decoder)	4
Embedding dimension	128
LAYERSCALE [65] initialization value	1.0
Optimizer	Ranger [68–70]
Learning rate	0.001
Batch size	1000
Codebook size	8192 (kin.) 32 768 (ext.)
α	10
β	0.9
ν	1
Replacement frequency	500 steps

A.4. VQ-VAE architecture changes

As the hybrid setup with continuous input and token-ID prediction requires on-the-fly decoding of token-IDs during the generation loop, the VQ-VAE has to be trained with a causal mask in the decoder. Additionally, we use the same transformer block architecture as in the backbone model, instead of the NormFormer [67] architecture that was used in OmniJet- α . The vector quantization is implemented using the `vqtorch` [37] library. The VQ-VAE hyperparameters are listed in table 2. As for the backbone, no dedicated hyperparameter tuning was performed, rather, a few configurations were tested and found to lead to stable results.

A.5. Training details

All models are trained with the Ranger [68] optimizer, which uses the Lookahead [69] optimizer with RAdam [70] as the inner optimizer. A constant learning rate of 10^{-3} and a batch size of 1000 is used for all trainings except for trainings where we use 10k jets or less, for which we use a batch size of 100. Pre-trainings are performed for a total of 1 M steps, with the final model state being used for evaluation. Classification trainings are performed for up to 1 M steps as well, but with early stopping based on the validation loss. Classification runs are repeated with five different random seeds and the average and standard deviation of the five runs is reported. The model state with the lowest validation loss is used for evaluation in classification runs.

Appendix B. Performance of the VQ-VAE

A comparison of both particle-level and jet-level features of the original JetClass dataset and the distributions obtained after tokenization (and subsequent decoding with the VQ-VAE decoder) are shown in figure 7. As can be seen in the distribution of the trajectory displacement d_0 , the VQ-VAE struggles to accurately reproduce this distribution. We found this feature to be very challenging to encode with a VQ-VAE, given that it features a very sharp peak at 0 with long tails towards small and large values.

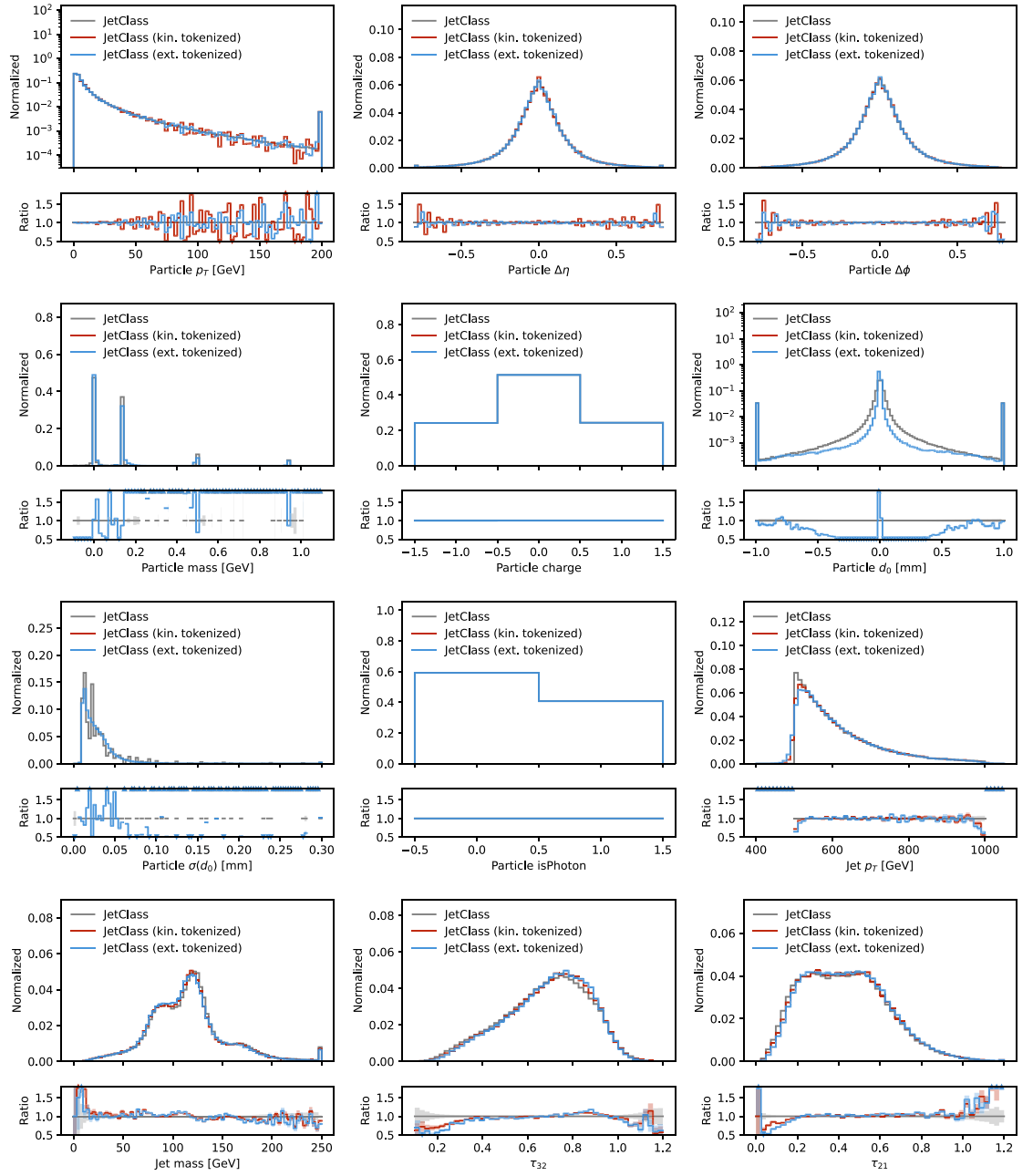


Figure 7. Comparison of the original JetClass dataset and the corresponding distributions obtained after tokenization. The tokenized datasets are the target for the generative models for the two feature sets investigated in this work: kinematics-only (kin.) and the extended feature set (ext.). Particle features related to trajectory displacement (d_0 and $\sigma(d_0)$) are only shown for charged particles.

However, we leave further studies regarding high-resolution tokenization of those features for future work.

Appendix C. Extending the feature set

To further improve the performance and utility of our models, we extend the input feature set of our model. We add the particle mass as well as the particle-ID and the trajectory displacement features to the input feature set of the model. A new VQ-VAE is trained for the extended feature set, using a code-book size 4 times larger than for the standard feature set, and a model is subsequently pre-trained with NTP.

The generative performance obtained with the two different input feature sets is shown in figure 8, which shows the particle p_T and two of the newly added features, the particle charge and the transverse

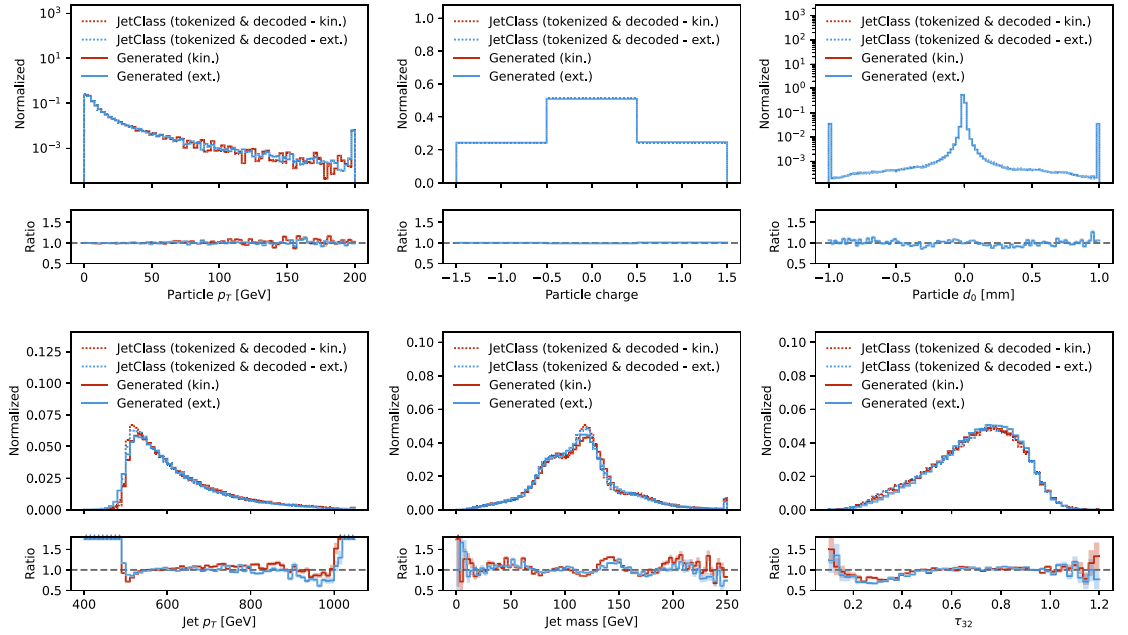


Figure 8. Comparison of the jets that are generated by the continuous-input model using the kinematics-only (kin.) and the extended feature set (ext.). The dotted lines show the distributions obtained from tokenization and subsequent decoding of the JetClass dataset with the corresponding VQ-VAE. The solid lines show the distributions obtained from the generative models. The ratio panels show the ratio between the distribution obtained from the generative model and the corresponding target distribution (which is shown in the same color in dotted linestyle).

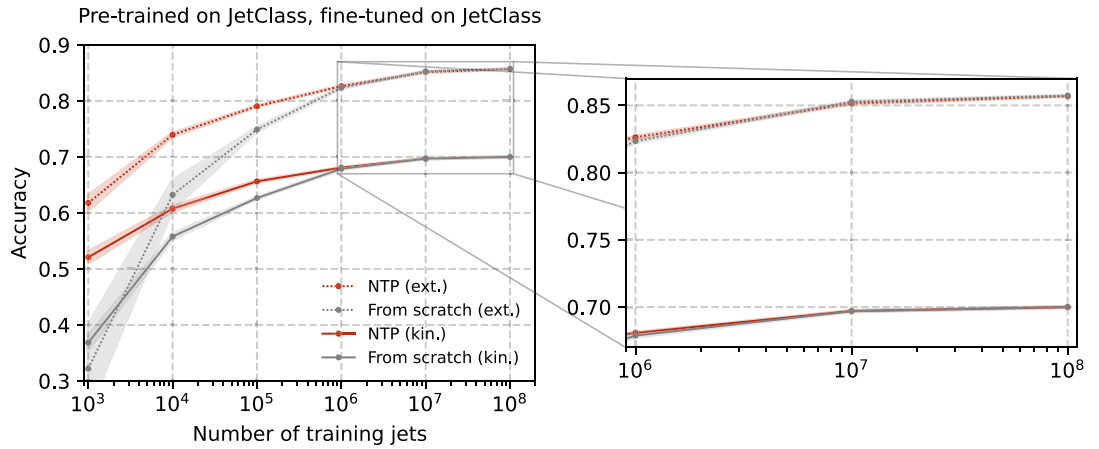


Figure 9. Multi-class classification performance on the JetClass dataset (all 10 jet types) as a function of the number of training jets comparing the different pre-training strategies with the corresponding from-scratch baselines for the kinematics-only (solid) and the extended feature set (dotted).

impact parameter d_0 , as well as several jet level features. The two dotted reference lines in figure 8 correspond to the different VQ-VAEs used for the two different feature sets, leading to slightly different target distributions during the generative training.

The model trained on the extended feature set shows similar performance on observables related to the kinematics of the particles (both on particle- and jet-level). The smoother distribution of the particle p_T in the extended feature set case arises from the VQ-VAE having a larger codebook size. Both the particle charge distribution and the d_0 distribution show good agreement with the corresponding target distribution.

By extending the input feature set, the classification performance is drastically improved as can be seen in figure 9. This overall improvement is expected as those features are known to be useful for jet tagging. The performance when trained on the whole JetClass dataset using the extended feature set

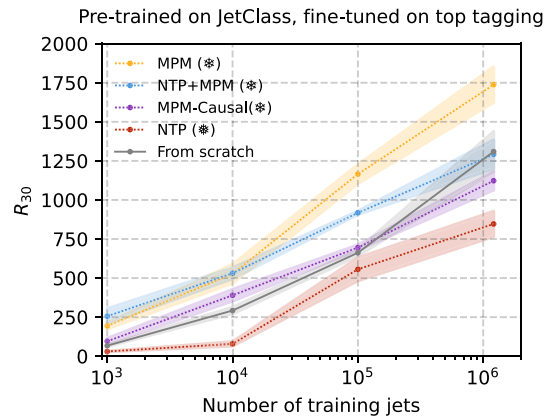


Figure 10. Classification performance on the top tagging dataset when fine-tuning with a fixed backbone.

leads to an accuracy of 85.7% in the baseline case whereas the kinematics-only model reaches an accuracy of only 70%. Generative pre-training leads to a notable improvement in classification performance for dataset sizes up to 1 M jets when using the extended feature set. For larger training dataset sizes, the performance of the pre-trained model converges to the performance of the from-scratch model.

Appendix D. Fine-tuning with fixed backbone representations

To investigate how aligned the pre-trained backbones are with the classification task, we perform classification fine-tuning where we only train the classifier head, shown in figure 10. While MPM and the joint pre-training perform better or on-par with the baseline model (in which also the backbone weights are allowed to change during training) for all dataset sizes, this is not true for MPM-Causal, which is outperformed by the from scratch model at a dataset size of 1.2 M training jets. Even more notable is the behavior for the model that is pre-trained with NTP: this model is outperformed for the whole training dataset range, meaning that this backbone representation does not offer an advantage over the baseline when the weights are not adapted during fine-tuning⁶.

In order to explore this behavior in more detail, we study the gradients, validation loss and validation R_{30} for the different setups. Figure 11(a) (left) shows the norm of the gradient of the backbone model for the individual runs from figure 10 at a dataset size of 10 000 jets. The backbone model obtained from NTP pre-training shows drastically larger gradients compared to the other methods. MPM, MPM-Causal and NTP+MPM have larger gradients than the from-scratch baseline (where the backbone is allowed to change) indicating that there remains significant gradient signal to update the frozen weights. However, those models still converge to a lower validation loss, as shown in figure 11(a) (middle). This is not true for the NTP model, which indicates that the representations obtained from this pre-training are not as aligned with the classification task as the ones of the other methods, or at least is less suited for subsequent fine-tuning to classification in a model with bi-directional attention model. We note that by switching from causal attention during pre-training to bi-directional attention during fine-tuning, the fixed representation is being used in a way that it was not trained for. And since it is fixed, it is not able to adapt. Figure 11(b) shows the corresponding behavior when using causal attention in the classifier training. The fine-tuning of the NTP-pre-trained model is significantly more stable, with the loss decreasing faster and smoother. Furthermore, the gradients show a similar behavior as the fine-tuning of the MPM-causal backbone. This supports the hypothesis above, that the directionality is an important component contributing to the performance loss. However, it is not a complete explanation as the fixed NTP backbone still lead to worse classification performance in this scenario. We hypothesize that the reason for this is that the output representation of the fixed NTP backbone will be too strongly aligned to the generative task making it more difficult to adapt to the classification task. In order to investigate whether this can be mitigated, we compare fine-tunings with both fixed and

⁶ This effect did not show up in the original OmniJet- α work, which used token-IDs as input rather than the continuous feature vector input used here. We believe that the baseline model in that case struggles with classification because it also needs to ‘decode’ the information contained in the token-IDs. The pre-training likely performs some of this decoding while training on the generative task, allowing it to outperform the baseline model. Using continuous input allows the classifier to completely bypass the token-IDs, making this baseline model more capable compared to that of [9].

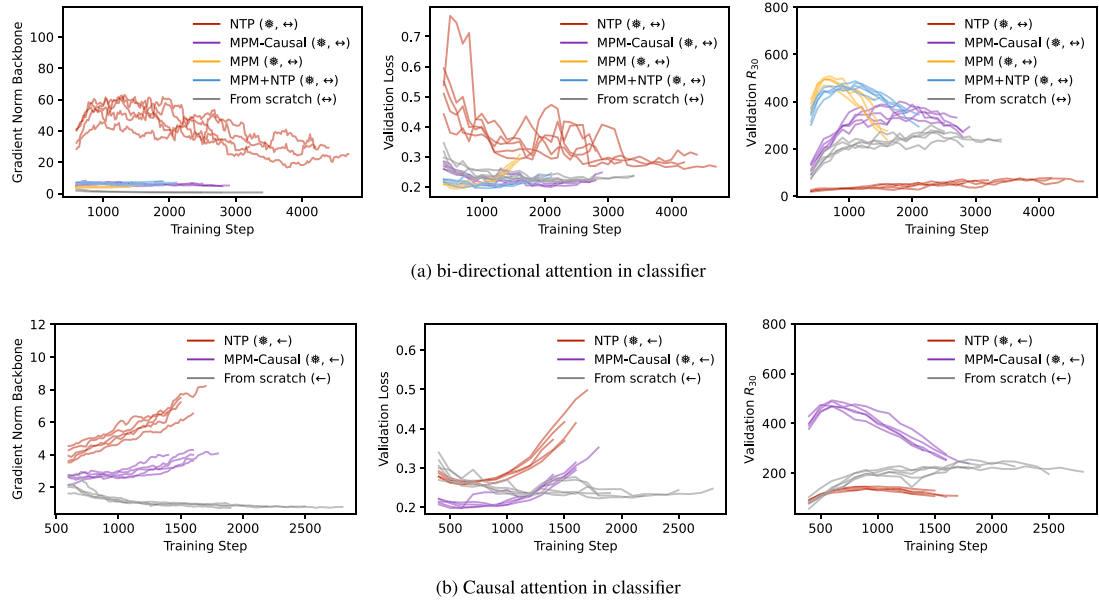


Figure 11. Evolution of the backbone's gradient norm, as well as the validation loss and R_{30} as a function of the training step. For each pre-training strategy five different runs are shown, corresponding to the five different runs with 10 000 training jets in figure 10. The snowflake symbol indicates that the weights of the backbone are fixed. The left-right arrow in the legend indicates that bi-directional attention is used in the backbone whereas a left arrow indicates that attention is only allowed to previous particles in the sequence (i.e. causal attention). The rolling average is shown for better readability. We show the average of the last 30 logged values of the gradients, which are logged every 20 training steps, and the average over the last 4 epochs for the validation loss and validation R_{30} .

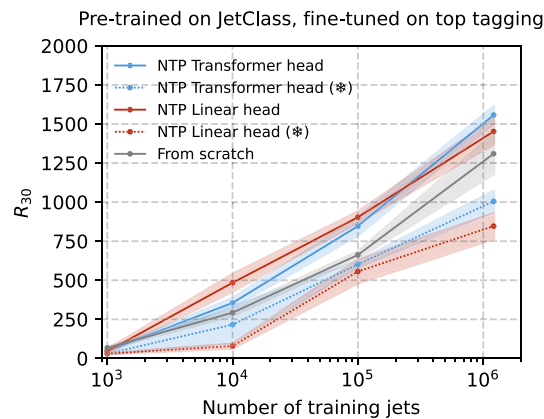


Figure 12. Classification performance on the top tagging dataset for NTP pre-trained models with two different architectures in the model head: *Linear head* corresponds to the model head being a single linear layer, whereas *Transformer head* corresponds to a model head with two transformer blocks and a linear layer.

trainable backbones for the default case where the NTP head is just a single linear layer to fine-tunings where the NTP head contains two transformer blocks and a subsequent linear layer like the MPM head. The classifier architecture is the same in all cases, only the head during pre-training differs between the different approaches. As shown in figure 12, while the performance with the fixed backbone improves, both pre-training architectures perform similarly in the subsequent fine-tuning when the backbone weights are allowed to change as well.

ORCID iDs

Joschka Birk  0000-0002-1931-0127

Anna Hallin  0000-0002-1551-814X

Gregor Kasieczka  0000-0003-3457-2755

Nikol Madzharova  0009-0008-4901-7926

Ian Pang  0000-0002-8225-7269

David Shih  0000-0003-3408-3871

References

- [1] Bommasani R *et al* 2021 arXiv:2108.07258
- [2] Hallin A 2025 (arXiv:2509.21434 [hep-ph])
- [3] Kasieczka G *et al* 2019 *SciPost Phys.* **7** 014
- [4] Golling T, Heinrich L, Kagan M, Klein S, Leigh M, Osadchy M and Raine J A 2024 *Mach. Learn. Sci. Tech.* **5** 035074
- [5] Leigh M, Klein S, Charton F, Golling T, Heinrich L, Kagan M, Ochoa I and Osadchy M 2025 *Mach. Learn.: Sci. Technol.* **6** 025075
- [6] Katel S, Li H, Zhao Z, Mokhtar F, Duarte J and Kansal R 2024 *Machine Learning and the Physical Sciences: Workshop at NeurIPS 2024* (arXiv:2412.05333 [hep-ph])
- [7] Wildridge A J, Rodgers J P, Colbert E M, yao Y, Jung A W and Liu M 2024 *38th Conference on Neural Information Processing Systems* (arXiv:2412.07867 [hep-ex])
- [8] Bardhan J, Agrawal R, Tilak A, Neeraj C and Mitra S 2025 arXiv:2502.03933 [cs.LG]
- [9] Birk J, Hallin A and Kasieczka G 2024 *Mach. Learn. Sci. Tech.* **5** 035031
- [10] Qu H, Li C and Qian S 2022 *Proc. 39th Int. Conf. on Machine Learning (Proc. of Machine Learning Research)* vol 162 eds K Chaudhuri, S Jegelka, L Song, C Szepesvari, G Niu and S Sabato (PMLR) pp 18281–92
- [11] Mikuni V and Nachman B 2025 *Phys. Rev. D* **111** L051504
- [12] Mikuni V and Nachman B 2025 *Phys. Rev. D* **111** 054015
- [13] Bhimji W, Harris C, Mikuni V and Nachman B 2025 arXiv:2510.24066 [hep-ph]
- [14] Ho J, Roberts B R, Han S and Wang H 2024 Pretrained event classification model for high energy physics analysis (arXiv:2412.10665 [hep-ph])
- [15] Harris P, Krupa J, Kagan M, Maier B and Woodward N 2025 *Phys. Rev. D* **111** 032010
- [16] Hao Z, Kansal R, Gandrakota A, Sun C, Jennifer N, Duarte J and Spiropulu M 2025 Rino: renormalization group invariance with no labels (arXiv:2509.07486 [hep-ex])
- [17] Vigl M, Hartman N and Heinrich L 2024 *Mach. Learn. Sci. Tech.* **5** 025075
- [18] Amram O, Anzalone L, Birk J, Faroughy D A, Hallin A, Kasieczka G, Krämer M, Pang I, Reyes-Gonzalez H and Shih D 2025 *Mach. Learn. Sci. Tech.* **6** 030601
- [19] Pang I, Faroughy D A, Shih D, Das R and Kasieczka G 2025 arXiv:2511.15779 [hep-ph]
- [20] Lv A, Zhang K, Xie S, Tu Q, Chen Y, Wen J-R and Yan R 2024 An analysis and mitigation of the reversal curse (arXiv:2311.07468)
- [21] BehnamGhader P, Adlakha V, Mosbach M, Bahdanau D, Chapados N and Reddy S 2024 LLM2Vec: large language models are secretly powerful text encoders (arXiv:2404.05961 [cs.CL])
- [22] Charpentier L G G and Samuel D 2024 GPT or BERT: why not both? (arXiv:2410.24159 [cs.CL])
- [23] Yu X, Guo B, Luo S, Wang J, Ji T and Wu Y 2024 Antlm: bridging causal and masked language models (arXiv:2412.03275 [cs.CL])
- [24] Liu H, Geng X, Lee L, Mordatch I, Levine S, Narang S and Abbeel P 2023 Towards better few-shot and finetuning performance with forgetful causal language models (arXiv:2210.13432 [cs.CL])
- [25] Gisserot-Boukhlef H, Boizard N, Faysse M, Alves D M, Malherbe E, Martins A F T, Hudelot C and Colombo P 2025 Should we still pretrain encoders with masked language modeling? (arXiv:2507.00994 [cs.CL])
- [26] Izdebski A, Olszewski J, Gawade P, Koras K, Korkmaz S, Rauscher V, Tomczak J M and Szczurek E, 2025 Synergistic benefits of joint molecule generation and property prediction (arXiv:2504.16559 [cs.LG])
- [27] Paszke A *et al* 2019 *Advances in Neural Information Processing Systems* vol 32 (Curran Associates, Inc.) pp 8024–35
- [28] Falcon W and The PyTorch Lightning team 2019 PyTorch lightning
- [29] Hunter J D 2007 *Comput. Sci. Eng.* **9** 90
- [30] T Pandas Development Team 2020 pandas-dev/pandas: Pandas
- [31] McKinney W 2010 *Proc. 9th Python in Science Conf.* eds S van der Walt and J Millman pp 56–61
- [32] Harris C R *et al* 2020 *Nature* **585** 357
- [33] Pivarski J, Elmer P and Lange D 2020 *EPJ Web Conf.* **245** 05023
- [34] Pivarski J, Osborne I, Ifrim I, Schreiner H, Hollands A, Biswas A, Das P, Roy Choudhury S, Smith N, Goyal M, Fackeldey P and Krommydas I 2025 Awkward array
- [35] Roy A, Pivarski J, Gray L, Papageorgakis C, Feickert M, Duarte J, Schreiner H, Kansal R, cmoore24 24, Sean github, Ho K W, Lieret K, Fackeldey P and Rothman S 2025 scikit-hep/fastjet
- [36] Cacciari M, Salam G P and Soyez G 2012 *Eur. Phys. J. C* **72** 1896
- [37] Huh M 2022 vqtorch: PyTorch package for vector quantization (available at: <https://github.com/minyoungg/vqtorch>)
- [38] Pedregosa F *et al* 2011 *J. Mach. Learn. Res.* **12** 2825
- [39] Buss T 2025 FLC-QU-hep/ranger-lite: RangerLite 1.0.0
- [40] Chopra S, Schreiner H, Rodrigues E, Eschle J and Pivarski J 2025 *J. Open Source Softw.* **10** 7791
- [41] Qu H, Li C and Qian S 2022 JetClass: A Large-Scale Dataset for Deep Learning in Jet Physics (1.0.0) [Data set] Zenodo (<https://doi.org/10.5281/zenodo.6619768>)
- [42] Kasieczka G, Plehn T, Thompson J and Russel M 2019 Top Quark Tagging Reference Dataset (v0 (2018_03_27)) [Data set] Zenodo (<https://doi.org/10.5281/zenodo.2603256>)
- [43] Alwall J, Frederix R, Frixione S, Hirschi V, Maltoni F, Mattelaer O, Shao H S, Stelzer T, Torrielli P and Zaro M 2014 *J. High Energy Phys.* **JHEP07(2014)079**
- [44] Sjostrand T, Mrenna S and Skands P Z 2008 *Comput. Phys. Commun.* **178** 852
- [45] Sjostrand T, Ask S, Christiansen J R, Corke R, Desai N, Ilten P, Mrenna S, Prestel S, Rasmussen C O and Skands P Z 2015 *Comput. Phys. Commun.* **191** 159
- [46] de Favereau J, Delaere C, Demin P, Giammanco A, Lemaître V, Mertens A and Selvaggi M DELPHES 3 2014 *J. High Energy Phys.* **JHEP02(2014)057**
- [47] Chatrchyan S *et al* CMS 2008 *J. Instrum.* **3** S08004
- [48] Cacciari M, Salam G P and Soyez G 2008 *J. High Energy Phys.* **04** 063
- [49] Aad G *et al* ATLAS 2008 *J. Instrum.* **3** S08003

- [50] van den Oord A, Vinyals O and Kavukcuoglu K 2017 *Advances in Neural Information Processing Systems* vol 30 eds I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan and R Garnett (Curran Associates, Inc.)
- [51] Huh M, Cheung B, Agrawal P and Isola P 2023 *Proc. 40th Int. Conf. on Machine Learning (Proc. of Machine Learning Research)* vol 202 eds A Krause, E Brunskill, K Cho, B Engelhardt, S Sabato and J Scarlett (PMLR) pp 14096–113
- [52] Buhmann E, Ewen C, Kasieczka G, Mikuni V, Nachman B and Shih D 2024 *Phys. Rev. D* **109** 055015
- [53] Devlin J, Chang M-W, Lee K, and Toutanova K 2019 BERT: pre-training of deep bidirectional transformers for language understanding (arXiv:1810.04805 [cs.CL])
- [54] He K, Chen X, Xie S, Li Y, Dollár P and Girshick R 2022 *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 16000–9
- [55] Bao H, Dong L, Piao S and Wei F 2022 *Int. Conf. on Learning Representations*
- [56] Wang R, Chen D, Wu Z, Chen Y, Dai X, Liu M, Jiang Y-G, Zhou L and Yuan L 2022 *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 14733–43
- [57] Baevisi A, Hsu W-N, Xu Q, Babu A, Gu J and Auli M 2022 *Proc. 39th Int. Conf. on Machine Learning (Proc. of Machine Learning Research)* vol 162 eds K Chaudhuri, S Jegelka, L Song, C Szepesvari, G Niu and S Sabato (PMLR) pp 1298–312
- [58] Wei C, Fan H, Xie S, Wu C-Y, Yuille A and Feichtenhofer C 2022 *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 14668–78
- [59] Spinner J, Bresó V, de Haan P, Plehn T, Thaler J and Brehmer J 2024 *38th Conference on Neural Information Processing Systems* (arXiv:2405.14806 [physics.data-an])
- [60] Brehmer J, Bresó V, de Haan P, Plehn T, Qu H, Spinner J and Thaler J 2025 *SciPost Phys.* **19** 108
- [61] Tschannen M, Eastwood C and Mentzer F 2024 arXiv:2312.02116 [cs.CV]
- [62] Birk J et al (available at: <https://github.com/uhh-pd-ml/enhancing-ntp4jets>)
- [63] Radford A, Narasimhan K, Salimans T and Sutskever I 2018 Improving language understanding by generative pre-training
- [64] Xiong R, Yang Y, He D, Zheng K, Zheng S, Xing C, Zhang H, Lan Y, Wang L and Liu T-Y 2020 *Proc. 37th Int. Conf. on Machine Learning (JMLR.org) ICML'20*
- [65] Touvron H, Cord M, Sablayrolles A, Synnaeve G and Jégou H 2021 *Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV)* pp 32–42
- [66] Darcet T, Oquab M, Mairal J and Bojanowski P 2024 *The Twelfth Int. Conf. on Learning Representations*
- [67] Shleifer S, Weston J and Ott M 2021 Normformer: improved transformer pretraining with extra normalization (arXiv:2110.09456 [cs.CL])
- [68] Wright L 2019 Ranger - a synergistic optimizer. (available at: <https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer>)
- [69] Zhang M, Lucas J, Ba J and Hinton G E 2019 *Advances in Neural Information Processing Systems* vol 32 eds H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox and R Garnett (Curran Associates, Inc.)
- [70] Liu L, Jiang H, He P, Chen W, Liu X, Gao J and Han J 2020 *Int. Conf. on Learning Representations*

Pre-Training for Simulation-Based Science: A Study on Jet Foundation Model Training Objectives

Ibrahim Elsharkawy^{&*}

*Department of Physics, University of Toronto and Vector Institute, Toronto, ON, Canada and
NERSC, Lawrence Berkeley National Laboratory, Berkeley, California, USA*

Joschka Birk^{&†} and Gregor Kasieczka[‡]

Institut für Experimentalphysik, Universität Hamburg, 22761 Hamburg, Germany

Vinicius Mikuni[§]

Nagoya University, Kobayashi-Maskawa Institute, Aichi 464-8602, Japan

Wahid Bhimji[¶]

NERSC, Lawrence Berkeley National Laboratory, Berkeley, California, USA

Benjamin Nachman^{**}

*Department of Particle Physics and Astrophysics,
Stanford University, Stanford, CA 94305, USA and
Fundamental Physics Directorate, SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA*

Foundation models (FMs) trained on large datasets and fine-tuned on downstream tasks have emerged as a powerful paradigm in AI for science. Industrial FMs are typically trained using self-supervision with masking due to the lack of labels. In many scientific domains, accurate simulations are plentiful and facilitate large, labeled datasets. This opens up new possibilities for pre-training. We present a systematic comparison of pre-training methods using the OMNILEARNED High Energy Physics FM framework. We test supervised classification, flow-matching generation, and self-supervised masked particle modeling. All models are pre-trained on the JetClass dataset and fine-tuned on two representative downstream tasks, top jet classification and JetNet conditional generation. Among other observations, for classification tasks, we find that pure classifier pre-training is optimal when downstream labels and model capacity are plentiful, but combining it with self-supervised masked particle modeling (MPM) is uniquely powerful in the low-finetuning label regime. Flow matching-based generative pre-training seems to provide little benefit for downstream classification, and interestingly, for downstream generation, we find that flow matching must be in the pre-training objective to see a significant finetuning advantage, hinting at the orthogonality of classification and generation tasks. That is, for a model to transfer to both generative and classification downstream tasks, it must be pre-trained on both. This study provides a template for controlled scaling analysis of pre-training objectives for foundation models in simulation-based sciences.

CONTENTS

I. Introduction	2
II. OMNILEARNED Review	3
III. Study Design	4
A. Pre-training Setup	4
1. Pre-training Objectives	4
2. Model Sizes and Training Hyperparameters	5
B. Finetuning Setup	6
1. Top Jet Classification	6
2. Jet Generation	6

* ibrahim.elsharkawy@mail.utoronto.ca

† joschka.birk@uni-hamburg.de

‡ gregor.kasieczka@uni-hamburg.de

§ vmikuni@hepl.phys.nagoya-u.ac.jp

¶ wbbhimji@lbl.gov

** nachman@stanford.edu

& Authors contributed equally.

IV. Results	7
A. Top Tagging	7
1. Finetuning trajectory over pre-training steps	7
2. Model size scaling	9
3. Scan over pre-training dataset size.	10
B. JetNet Generation	11
1. Scan over dataset size	14
2. Performance vs. training duration	16
V. Conclusion	17
Code Availability	18
Acknowledgments	18
References	19
A. OMNILEARNED Modules	20
B. Hyperparameters	21
1. Pre-training Hyperparameters	21
2. Top Tagging Hyperparameters	21
3. Generation Hyperparameters	21
C. More Top Jet Classification Results	22
D. More Jet Generation Results	23

I. INTRODUCTION

The foundation model paradigm, pre-training a large model on a broad dataset and fine-tuning on specific downstream tasks, has recently been adopted for various AI for science applications [1–16]. Despite this progress, fundamental questions remain. Namely, which pre-training objective transfers best to which downstream task, and how strongly does this depend on the amount of pre-training data, downstream data, and model capacity? AI for science often has access to large simulated datasets, with labels available at scale. How does pre-training in a supervised fashion compare to self-supervised and generative objectives?

In this work, we attempt to address these questions using the OMNILEARNED framework [1–3]. OMNILEARNED is a foundation model in high-energy physics (HEP) trained on jets, variable-number particle sprays that arise from the showering and subsequent hadronization of free partons in high-energy colliders. Parton-shower and detector simulators such as PYTHIA, HERWIG, DELPHES, and GEANT4, and others, reproduce real collider data with a fidelity that is likely unrivaled among scientific domains, as validated by precision measurements at the LHC [17–21]. Importantly, datasets at the $\sim 10^9$ scale, including JetClass [22] and the billion-jet dataset released with OMNILEARNED [1], are already publicly available, removing potential data-scaling limitations. Jets are also more compact than other data types targeted by foundation models, with $\mathcal{O}(100)$ particle constituents per example rather than the tens of thousands of pixels found in image/video data. This makes a full sweep across pre-training objectives, dataset sizes, and model capacities computationally tractable. Despite the compactness, jet data is complex and requires data and model scale for downstream tasks. Published scaling-law studies for jet classification and generation [1, 5, 23–27] show smooth, predictable performance gains with both model size and pre-training-data up until at least $\sim 10^9$ jets and parameters. Further, the downstream tasks we consider (jet classification and jet generation) are not toy benchmarks but useful to ongoing analyses at the Large Hadron Collider (LHC) and beyond: jet tagging is an essential component of LHC physics analyses [28] and generative models for jet constituents show promising potential for anomaly detection efforts [1–3, 29, 30]. Finally, jets have been used as a model for data in AI interpretability studies [31, 32], and the OMNILEARNED model has shown cross-domain transfer to other physical domains [33–35].

Given the setup, we pre-train a set of variants of OMNILEARNED using the following objectives,

1. **Jet classification:** supervised cross-entropy on jet type labels denoting the initial particle.
2. **Jet generation:** flow-matching with a velocity-prediction objective.

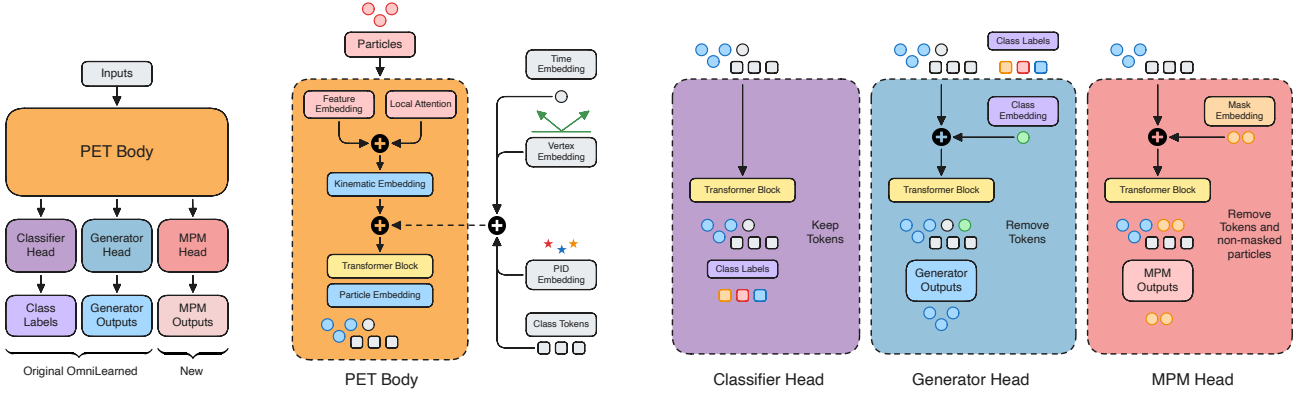


FIG. 1. Illustration of the model architecture used in this work. The generator and classifier components were already used in the original OMNILEARNED [1] work. The additional MPM [14, 15] head is added to the OMNILEARNED framework here.

3. Masked Particle Modeling (MPM): a self-supervised objective predicting masked jet constituents.

We also train models with all pairwise combinations (Classifier+Generator, Classifier+MPM, Generator+MPM) and all three. For selected configurations, we also include *perturbed* classification losses on noised or masked inputs, promoting robustness of the learned representations. We vary the three axes of pre-training dataset size, finetuning dataset size, and model size. Finetuning is performed on two downstream tasks: top-quark jet tagging (a classification benchmark) and JetNet conditional generation (a generative benchmark). This study provides a template for controlled scaling analysis of pre-training objectives for jet foundation models.

The paper is organized as follows. Section II reviews the OMNILEARNED architecture and presents the architecture changes adopted in this work. Section III describes the study design, including the pre-training configurations in Sec. III A and the finetuning protocol in Sec. III B. Section IV presents results on top tagging (Sec. IV A) and JetNet generation (Sec. IV B), and conclusions are provided in Sec. V.

II. OMNILEARNED REVIEW

OMNILEARNED [1] is a point-edge transformer (PET) foundation model for jet physics and an upgrade of the original OMNILEARN [2, 3]. OMNILEARNED was pre-trained on $\sim 10^9$ jets drawn from JetClass, JetClass 2, ATLAS Top Tagging, H1 DIS, Aspen Open Jets, and CMS QCD/BSM [1, 5, 22, 36–38]. The model consists of a shared body that produces per-particle and per-jet representations, and task-specific heads that can be attached or removed depending on the training objective. This design enables multi-task pre-training where the body is shared across all objectives. A diagram of the model architecture used in this study can be found in Figure 1. The architecture follows the original OMNILEARNED architecture, with the addition of another model head. The original OMNILEARNED framework was pre-trained on the joint classification + generation objective [1–3]. To extend the comparison to fully self-supervised pre-training, we introduce a masked particle modeling (MPM) [14, 15] head. The remaining architecture is unchanged.

Two defining aspects of OMNILEARNED’s PET are a local embedding block and a physics-informed attention bias, which are described in App. A. Due to unique architecture choices and extensive pre-training, OMNILEARNED has shown great performance and flexibility in jet-related tasks (jet classification, generation, anomaly detection, unfolding) and cross-domain point cloud tasks (neutrino physics, molecular dynamics, and cosmology) [33–35].

a. Input features Each jet is represented as a variable-length set $\{x_i\}_{i=1}^N$ of $< \sim 150$ constituents. The input is the four-vector $x_i = (\Delta\eta_i, \Delta\phi_i, \log p_{T,i}, \log E_i)$, with $\Delta\eta_i, \Delta\phi_i$ measured relative to the jet axis and $\log p_{T,i}$ computed relative to the beam axis. When available, particle identification (PID) is embedded through a learned lookup table, and charged-particle vertex information is encoded by a two-layer MLP. These optional embeddings are *added* to the kinematic embedding, and are set to zero when the information is not present in a dataset. The flow-matching time $t \sim \mathcal{U}(0, 1)$, encoded with Fourier features followed by an MLP, is appended to the point cloud as an additional token.

b. Classifier and generator head The classifier head takes the $N_{\text{tok}} = 4$ global tokens along with the input embeddings produced by the body, passes them through additional attention blocks. The global tokens are then flattened and passed through an MLP, and outputs C class logits $\hat{y}$. The classifier is trained with standard cross-

entropy loss against the true class label y ,

$$\mathcal{L}_{\text{class}} = - \sum_{c=1}^C y_c \log \hat{y}_c. \quad (1)$$

The generator head is trained with flow matching [39] for conditional jet generation. Given a clean jet x and a time $t \sim \mathcal{U}(0, 1)$, the head receives the perturbed input $\varphi(t) = (1-t)x + t\epsilon$ with $\epsilon \sim \mathcal{N}(\mu, \sigma)$ and predicts the velocity field $\frac{d}{dt}\varphi(t) = \epsilon - x$. The generator is trained with a mean squared error against the true velocity field,

$$\mathcal{L}_{\text{gen}} = \left\| \hat{v} - \frac{d}{dt}\varphi(t) \right\|^2. \quad (2)$$

c. Masked Particle Modeling (MPM) head The MPM head follows a regression-based paradigm as introduced in [14, 15]. A fixed fraction of the input particles (40% in this case) is masked before passing the point cloud to the body. The learned representation of the survived (=not masked) particles is then used as context to reconstruct the feature vector of the masked particles. Masked particles are re-introduced to the particle set after the body. This is implemented with learnable mask tokens that encode partial information about the p_T of the masked particles. This head consists of two transformer blocks, followed by a linear layer. Only the kinematic features of the masked particles are predicted by the MPM head, following the same paradigm as in the generator head. The MPM loss is given by,

$$\mathcal{L}_{\text{MPM}} = \frac{1}{|\mathcal{M}|} \sum_{\mathcal{M}} \|\hat{x}_i - x_i\|, \quad (3)$$

where $\mathcal{M}$ is the set of masked particle indices and $\hat{x}_i$ is the head's prediction.

III. STUDY DESIGN

This section describes how we use the OMNILEARNED framework to study how the choice of pre-training objective influences downstream fine-tuning performance. The architecture is held fixed, only the set of active tasks heads, their associated losses, and the size of the body are varied. The first subsection (Sec. III A) describes the pre-training configurations, the second (Sec. III B) describes the fine-tuning protocol used to evaluate them.

A. Pre-training Setup

Our goal is to benchmark supervised, generative, and self-supervised pre-training objectives and combinations of those objectives as a function of the downstream task, fine-tuning dataset size, model size, and pre-training dataset size. Given the three model heads with their dedicated task, this yields three pre-training objectives that can be activated independently or in combination, giving seven pre-training modes. We pre-train each configuration at three model sizes and, for the smallest size we additionally vary the number of pre-training examples to study how each objective scales with data.

The pre-training is restricted to the publicly available JetClass dataset [22, 40] throughout the study. JetClass contains jets across ten physically distinct classes generated with MADGRAPH5 [41], showered, and hadronized with PYTHIA 8 [17], and passed through the DELPHES 3.4.3 detector simulation with the CMS card [19]. We use the standard 100M/5M/20M training-validation-test split. The use of a single labeled dataset ensures that all pre-training objectives have access to identical input data, the supervised classifier fit to the JetClass labels, the generation objectives are conditioned on them, and the self-supervised objective ignores them (only trained on the per-particle kinematic and additional features).

1. Pre-training Objectives

The total pre-training loss is a sum over the active head losses, and in the case of all heads being active,

$$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{MPM}} + \alpha(t)^2 \mathcal{L}_{\text{class}}^{\text{gen, perturb}} + w_{\text{MPM}} \mathcal{L}_{\text{class}}^{\text{MPM, perturb}}, \quad (4)$$

TABLE I. The seven pre-training configurations studied in this work.

Configuration	Pre-training loss
Classifier	$\mathcal{L} = \mathcal{L}_{\text{class}}$
Generator	$\mathcal{L} = \mathcal{L}_{\text{gen}}$
MPM	$\mathcal{L} = \mathcal{L}_{\text{MPM}}$
Classifier + Generator	$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{gen}} + \alpha(t)^2 \mathcal{L}_{\text{class}}^{\text{gen, perturb}}$
Classifier + MPM	$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{MPM}} + w_{\text{MPM}} \mathcal{L}_{\text{class}}^{\text{MPM, perturb}}$
Generator + MPM	$\mathcal{L} = \mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{MPM}}$
Classifier + Generator + MPM	$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{MPM}} + \alpha(t)^2 \mathcal{L}_{\text{class}}^{\text{gen, perturb}} + w_{\text{mpm}} \mathcal{L}_{\text{class}}^{\text{mpm, perturb}}$

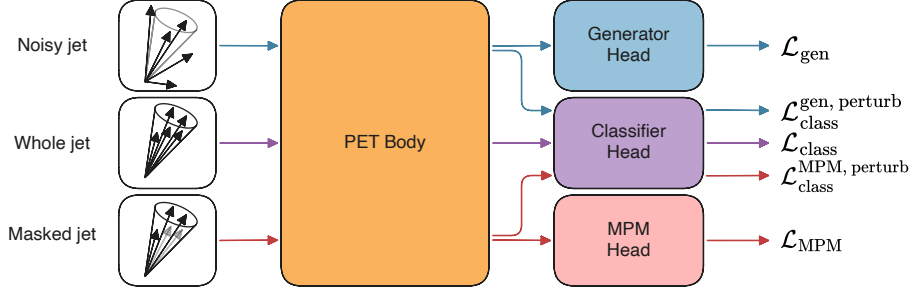


FIG. 2. Illustration of the different jet augmentations for the different tasks and how the information flows through the model in the forward pass.

where each term is included only when the corresponding head is active in a given pre-training run. In addition to the three primary head losses, two perturbed-input classification losses appear in Eq. 4. These terms are activated whenever the classifier head is active with the generator and/or the MPM head. These terms originate from passing a perturbed body representation (noised for the flow-matching forward process, or masked for MPM) through the classifier head, and adding a cross-entropy loss of its prediction against the true label to the total loss. $\alpha(t) \equiv 1 - t$ is the interpolation schedule and w_{MPM} is set to $1 - f_m$ where f_m is the fraction of masked particles ($f_m = 0.4$ in our study). The main loss components of the individual tasks enter the total loss with equal weight. Investigating weighted combinations of the different tasks is left for future studies. An illustration of the different forward passes is presented in Figure 2.

The seven pre-training configurations studied in this work are summarized in Table I, where the Classifier+Generator configuration corresponds to the original OMNILEARNED pre-training objective [1, 2].

2. Model Sizes and Training Hyperparameters

We train each of the seven configurations at three model sizes, summarized in Table II. The local-attention block, the input embedding of Eq. A2, the interaction bias of Eq. A1, and the head architectures are held fixed across sizes. The body depth, the embedding dimension, and the number of attention heads varies with model size.

Because the micro model is small enough to train to convergence rapidly, for the Micro model only, we vary the number of pre-training jets N_{pre} over a log sweep from $N_{\text{pre}} = 10^5$ to the full 10^8 JetClass training set.

a. Pre-training hyperparameters and schedule Pre-training proceeded on NERSC Perlmutter A100x4 nodes [42]. All pre-training runs use a fixed global batch size of 8192 jets, kept constant across model sizes. The medium models, and all models in which the MPM head is active (regardless of size), are trained with the Lookahead optimizer [43] with RAdam [44] as the inner optimizer, referred to as RANGER [45, 46]. All remaining configurations are trained with the LION [47] optimizer matching the original OMNILEARNED [1]. We use a cosine learning-rate schedule with a linear warmup phase. All remaining hyperparameters were tuned independently for each model size, pre-training objective pair to maximize that configuration's fine-tuning performance. Training was terminated when either the validation loss plateaued or began to overfit, or downstream finetuning performance ceased to improve. All hyperparameters not mentioned but used in this study are laid out in App. B.

TABLE II. Model configurations used in this study. N_{body} is the number of transformer blocks in the body, d_{base} is the embedding dimension, and N_{heads} is the number of attention heads. N_{nodes} is the number of NERSC A100x4 compute nodes used during pre-training.

	N_{body}	d_{base}	N_{heads}	Params	N_{nodes}	Local Batch Size
Micro	3	32	4	$\sim 100\text{k}$	4	256
Small	8	128	8	$\sim 7\text{M}$	8	128
Medium	12	512	16	$\sim 51\text{M}$	32	32

B. Finetuning Setup

Each pre-trained body is evaluated on two downstream tasks: a standard top vs. QCD jet classification task, and a standard jet generation task. For every pre-trained checkpoint we initialize the model from the pre-training weights and train on the downstream task, including all weights of pre-training task heads whose shapes are compatible with the downstream task head. For both tasks, we additionally train models from scratch as a reference.

1. Top Jet Classification

For top-jet classification we use the publicly available Top Quark Tagging Reference Dataset [48, 49], the same benchmark used to evaluate the original OMNILEARNED [1, 2]. The dataset contains top jets (produced via a weak decay $t \rightarrow W^+b \rightarrow q\bar{q}'b$ and QCD (produced via one free parton) generated with PYTHIA 8 and passed through the DELPHES fast detector simulation, and is split into 1.2M training, 400k validation, and 400k test jets. All final results reported in this work are computed on the test set.

We initialize each finetuning run from one of the pre-trained checkpoints described in Sec. III A. All body and embedding weights are loaded, and the pre-trained classifier head is also loaded when available. The classifier output layer is reinitialized to match the binary output of the task, and unused task heads (generator and MPM) are discarded.

Finetuning uses the Lion optimizer [47] with a learning rate and weight decay chosen separately for each model size to maximize validation performance. The from-scratch baseline randomly initializes the entire model and is trained with a separately tuned learning rate and weight decay. For finetuning we use a linear warmup followed by a cosine decay. Early stopping on the validation loss is applied throughout. To map out the effect of increasing the finetuning dataset size and efficiency of each pre-training objective, we finetune over a logarithmic sweep of the number of available jets, $N_{\text{ft}} \in \{10^3, 10^4, 10^5, 1.2 \times 10^6\}$, where the last point uses the entire training set.

For each pre-training configuration, model size, N_{ft} combination, finetuning is performed *three times* with different random seeds for the data subset selection and weight initialization of fresh layers, and we report the mean and standard deviation across the three runs. After finetuning, each model is evaluated on the held-out test set. We report standard top-tagging metrics as a function of the pre-training configuration, the model size, and the number of finetuning examples N_{ft} .

2. Jet Generation

The generative downstream task is evaluated on the public JetNet dataset [50, 51]. This dataset contains jets originating from gluons, light quarks, W -bosons, Z -bosons and top quarks produced in proton-proton collisions at a center-of-mass energy of 13 TeV and carry a transverse momentum of $p_{\text{T}}^{\text{jet}} \approx 1$ TeV. The used dataset split corresponds to 550k jets for model training, 60k jets for validation, and 260k jets for testing. The maximum number of available particles stored in the dataset (150) is used. Only kinematic features of the jet constituents are available in the JetNet dataset and the approximation $m \approx 0$ is used to calculate the particle's energy for the input features. While the generative pre-training component is conditioned only on the jet type, we further add the jet mass and the jet transverse momentum as conditioning features. The jet type information enters the model after the backbone, whereas the jet mass and the jet p_{T} information is fed to the body in the form of an additional particle. In our study here, we use the jet-level values from the test dataset as conditioning features during generation. In an actual application of this generative model those would be generated by another model for jet-level features.

We evaluate the generative downstream task for three downstream dataset sizes $N_{\text{JetNet}} \in \{10^4, 10^5, 5.5 \times 10^5\}$. When initialized from a pre-trained checkpoint, the weights of the pre-trained body are loaded. If the pre-training includes the generative task they corresponding weights of the generator head are loaded as well. As in the top tagging downstream task, the last layer of the generator head is re-initialized even when the remaining weights are loaded from

a pre-trained checkpoint. In addition to that, the jet type embedding at the beginning of the generator head is always re-initialized as well, as the jet type mapping differs between pre-training and downstream task. The trainings on the generative task are performed with the Lion [47] optimizer, training each configuration for 70k training steps with a global batch size of 4096, corresponding to a maximum of 520 epochs when the largest dataset size of 5.5×10^5 is used. A cosine learning rate schedule with linear warmup is used. The final epoch is used for evaluation, unless overtraining occurs in which case the checkpoint corresponding to the lowest smoothed validation loss is used. The limited training duration of 70k training steps on the generative task is chosen as this was found to be a suitable training duration for the small model size. Initial tests of longer trainings with the medium model size showed that further training can improve the generative performance of those models. Yet, we stick with a maximum of 70k training steps for all generative trainings due to the computational constraints imposed by the large number of configurations studied in this work.

IV. RESULTS

Below, we summarize our findings for the two downstream tasks (top-jet classification and jet generation). Other metrics not reported in this section can be found in App. C and App. D.

A. Top Tagging

We run the finetuning setup described in Sec. III B 1 and report the standard top-tagging metric background rejection $1/\epsilon_{\text{bg}}$ at a signal efficiency of $\epsilon_s = 0.3$ used in the literature [1, 2, 48]. We plot the number of finetuning examples (the number of training jets taken from the top tagging dataset) vs. performance for various model sizes in Figs. 3 a-c. In these, we see that at $N_{\text{ft}} = N_{\text{max}} \equiv 1.2 \times 10^6$, pure classifier pre-training is the single best configuration at every model size when using the best pre-train validation loss checkpoint for fine-tuning. As we will see in Sec IV A 1, this does not necessarily hold when looking at the entire pretraining trajectory.

The low-data regime is qualitatively different. To summarize each pre-training configuration, model size, N_{top} combination with a single number in the low data regime, we define the data-efficiency score,

$$\text{DES}(N_{\text{top}}) = \frac{\text{Perf}_{\text{pre-trained}}(N_{\text{top}})}{\text{Perf}_{\text{scratch}}(N_{\text{max}})} \times \frac{N_{\text{max}}}{N_{\text{top}}}, \quad (5)$$

where Perf is again background rejection at fixed signal efficiency $\epsilon_s = 0.3$, N_{top} is the number of finetuning examples used during finetuning, and $N_{\text{max}} = 1.2 \times 10^6$ is the full top-tagging training set. A value $\text{DES}(N_{\text{top}}) \gg 1$ indicates that pre-training provides an advantage compared to a model trained from scratch.

At $N_{\text{top}} = 10^4$ shown in Fig. 4, Classifier+MPM is the top configuration at *every* model size. Pure classifier pre-training is a clear second at Small and Medium but drops to fifth at Micro, where it is beaten by both Classifier+MPM and the fully self-supervised standalone MPM head. The Classifier+Generator+MPM configuration tracks Classifier+MPM closely across all three sizes, indicating that the gain over classifier-only comes from the masked-particle term rather than from generation.

Two additional patterns stand out. First, all classifier-containing multi-objective configurations converge to within a few percent of each other at $N_{\text{ft}} = N_{\text{max}}$ within uncertainties, which is not the case for single objective pre-training. Second, pure generator pre-training is the weakest performing configuration throughout, at every N_{top} , model size pair it either matches or underperforms the from-scratch baseline. Generator+MPM inherits this pattern in the low-data regime, suggesting that flow-matching alone fails to teach the body's representation in a way that transfers usefully to classification, and that combining it with MPM only dilutes the masked-particle signal. The pattern of purely generative pre-training being outperformed by MPM pre-training when fine-tuned on top tagging was previously seen in [6], where the generative task is implemented with next token prediction.

Taken together, these observations suggest that when both the model and the labeled downstream sample are large, supervised classifier pre-training performs the best for classification downstream tasks. When either is limited, combining classifier pre-training with the self-supervised MPM head yields the largest improvement.

1. Finetuning trajectory over pre-training steps

To probe how finetuning performance evolves as a function of pre-training steps, we finetune a sequence of pre-training checkpoints for the Classifier and Classifier+Generator configurations at the Medium model size. We see

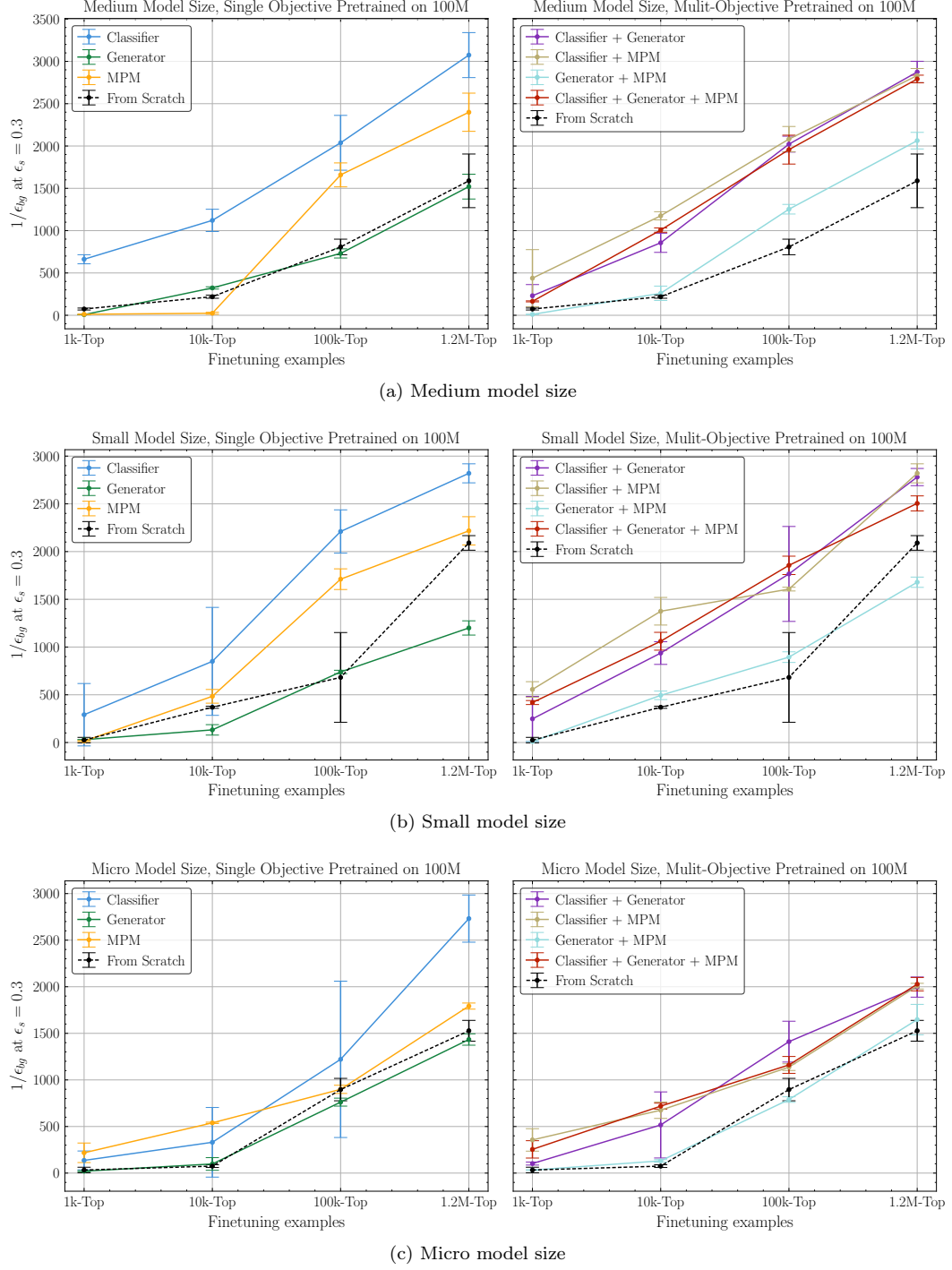


FIG. 3. Top-tagging background rejection at $\epsilon_s = 0.3$ vs. top-tagging dataset size for different model sizes (rows), pre-trained on 100M JetClass jets. Single pre-training losses are shown on the left while combinations of multiple losses are shown on the right.

that more pre-training is not always better, both configurations reach a peak in downstream performance well before pre-training converges, after which top-tagging $1/\epsilon_{bg}$ either plateaus or decreases (Fig. 5). Adding Generation seems to regularize this effect. Pearson coefficients quantifying the correlation between the two metrics and pre-training validation loss are given in Table III and show strong but not perfect correlation.

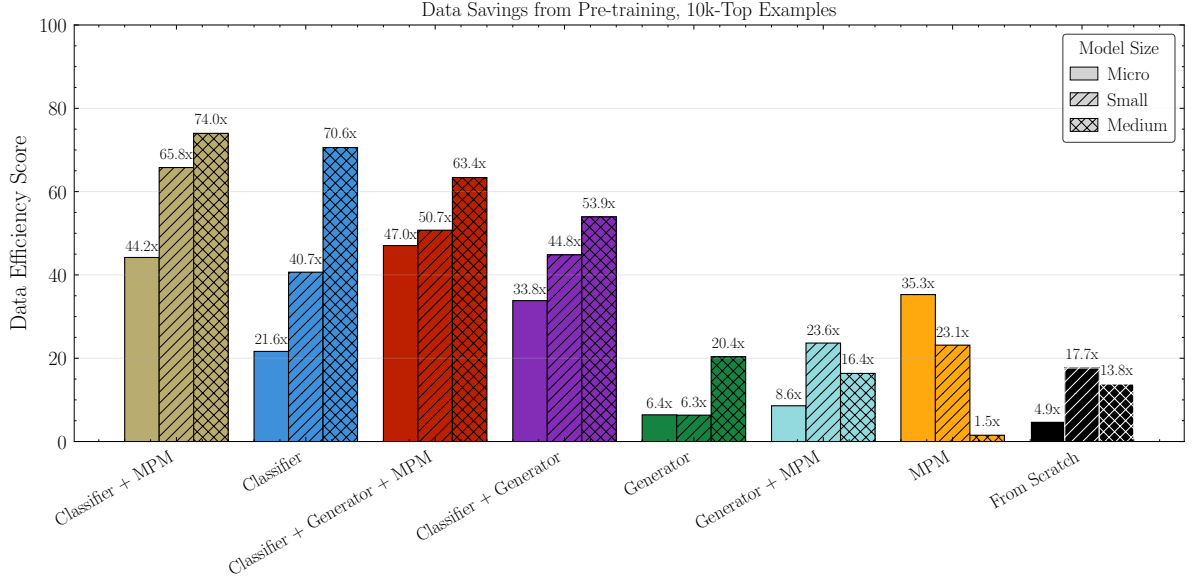


FIG. 4. Data-efficiency score $\text{DES}(N_{\text{top}})$ of Eq. (5) at $N_{\text{top}} = 10^4$ across the seven pre-training configurations and all three model sizes.

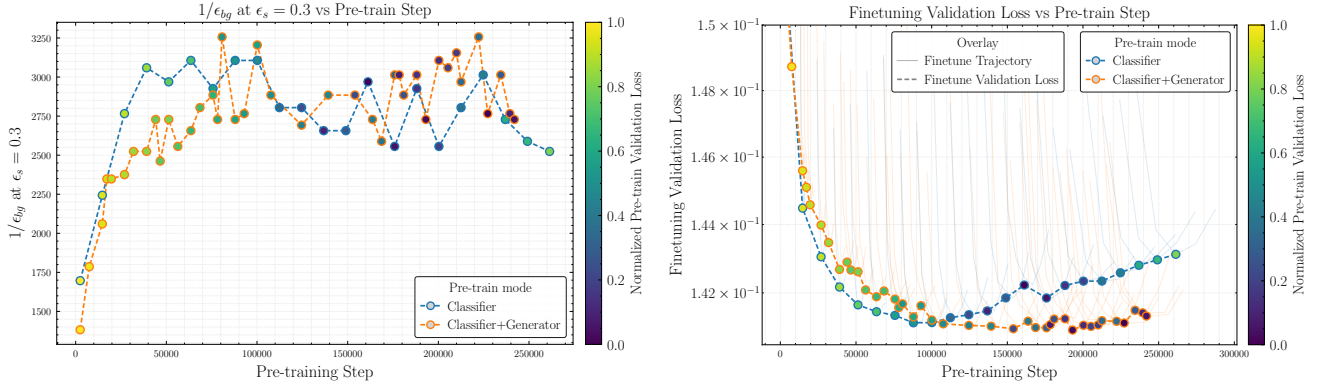


FIG. 5. Downstream metrics vs. pre-training step for the Classifier and Classifier+Generator configurations at the Medium model size, fine-tuned on the full $N_{\text{ft}} = N_{\text{max}}$ top-tagging set. Left gives background rejection $1/\epsilon_{\text{bg}}$ at $\epsilon_s = 0.3$. Right gives finetuning validation loss, with overlaid finetuning trajectories per checkpoint. Here, only one finetuning with one seed is performed. Marker color encodes the normalized pre-training validation loss. Results in Fig 3 correspond to finetuning checkpoint with the lowest *pre-training validation loss*. Raw pre-training and finetuning metrics as a function of pre-train step can be found in App. C in Table XVIII and Table XIX.

TABLE III. Pearson correlation r between pre-training validation loss and two downstream metrics across the sequence of pre-training checkpoints shown in Fig. 5. N is the number of checkpoints saved along the trajectory, plotted in Figure 5.

Configuration	N	$r(\mathcal{L}_{\text{pre-train}}, \mathcal{L}_{\text{FT}})$	$r(\mathcal{L}_{\text{pre-train}}, 1/\epsilon_{\text{bg}})$
Classifier	22	+0.962	−0.751
Classifier + Generator	40	+0.896	−0.765

2. Model size scaling

Figure 6 shows the full top tagging data ($N_{\text{ft}} = N_{\text{max}}$) background rejection as a function of model size, in $\log(N_{\text{params}})$, for all seven pre-training configurations and the from-scratch baseline. Every pre-training configuration improves with size, but to very different degrees, and Table IV quantifies this with a linear fit of $1/\epsilon_{\text{bg}}$ vs $\log(N_{\text{params}})$.

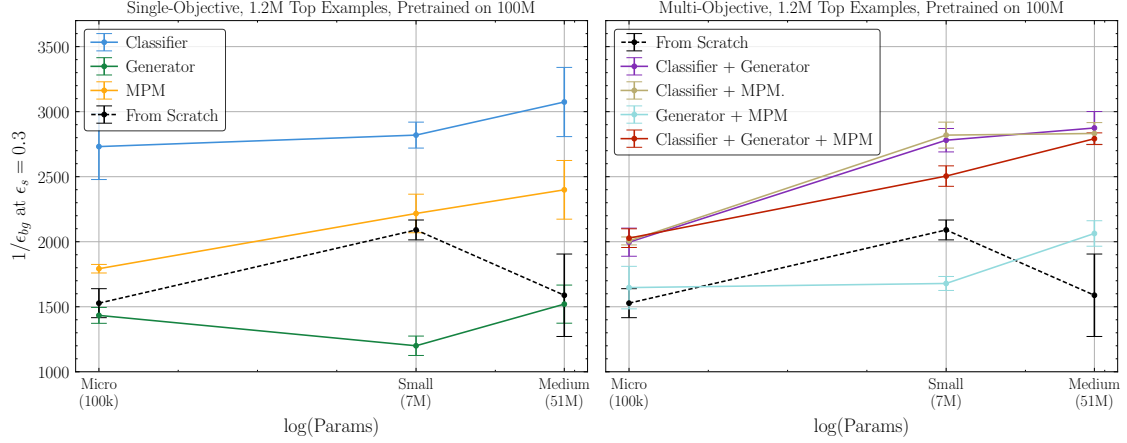


FIG. 6. Top-tagging background rejection at $\epsilon_s = 0.3$ vs model size for the seven pre-training configurations at $N_{\text{ft}} = N_{\text{max}}$.

TABLE IV. Linear fits of background rejection $1/\epsilon_{\text{bg}}$ at $\epsilon_s = 0.3$ vs $\log(N_{\text{params}})$. Slope m gives the performance gain per parameter count, R^2 measures the linearity of the trend. Bold represents the largest slope (best scaling).

Configuration	m	R^2
Classifier	114.2	0.787
Generator	6.9	0.003
MPM	225.0	1.000
From Scratch	67.2	0.090
Classifier + Generator	340.2	0.953
Classifier + MPM	326.6	0.910
Generator + MPM	132.1	0.623
Classifier + Generator + MPM	278.6	0.996

The from-scratch baseline and the pure-Generator configuration are essentially flat, confirming that flow-matching pre-training alone fails to deliver usable scaling. Generator+MPM shows similar behavior. On the other hand, the three classifier-containing multi-objective configurations have the steepest slopes. Interestingly, they converge to within $\sim 1\%$ of each other at the Medium scale, indicating that once capacity is sufficient, the choice of *which* self-supervised head accompanies the classifier loss may have little effect on the final performance.

Standalone MPM is the most striking single-objective result, its slope is the largest among single-objective configurations and nearly $2\times$ that of the pure Classifier, suggesting that the masked-particle objective alone, with no supervised signal, scales more efficiently with capacity than the supervised classifier loss, even though the supervised loss yields higher absolute performance at every model size studied. We leave scaling models to larger sizes for future work, where it would be interesting to see if the MPM, and Classifier+Generator+MPM continue to scale beyond other configurations.

3. Scan over pre-training dataset size.

To probe how each pre-training objective responds to the amount of JetClass data, we sweep the pre-training sample size $N_{\text{pre}} \in \{10^5, 10^6, 10^7, 10^8\}$ for the Micro model under all seven pre-training configurations and finetune at two top-tagging dataset sizes, $N_{\text{top}} = 10^4$ and $N_{\text{ft}} = N_{\text{max}}$ (Fig. 7). Table V gives the slope m and R^2 of a linear fit of $1/\epsilon_{\text{bg}}$ vs $\log(N_{\text{pre}})$ for each configuration and each N_{top} .

At the high N_{top} (shown in the right panel of Fig. 7), every configuration improves substantially with N_{pre} , with an average slope across pre-training modes of $\bar{m} = 182.5$ and an average $\bar{R}^2 = 0.807$. Pure Classifier pre-training dominates this regime, its slope is over $2\times$ the next-best configuration and yields a near-perfect linear scaling, in clear contrast to the much shallower scaling of the Generator-only and Generator+MPM configurations. In short, when downstream labels are abundant, more pre-training data gives better performance, and the uplift is the most for the purely supervised objective.

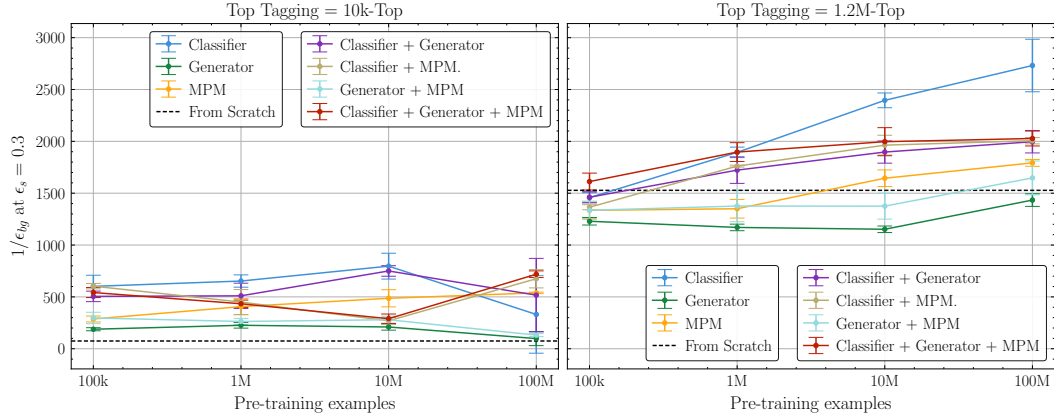


FIG. 7. Background rejection $1/\epsilon_{bg}$ at $\epsilon_s = 0.3$ vs pre-training dataset size N_{pre} for the micro model under all seven pre-training configurations. $N_{ft} = 10^4$ is shown on the left. $N_{ft} = N_{max} = 1.2 \times 10^6$ is shown on the right. The from-scratch baseline (dashed) does not depend on N_{pre} .

TABLE V. Linear fits of $1/\epsilon_{bg}$ at $\epsilon_s = 0.3$ vs $\log(N_{pre})$ for the Micro model, at two fine-tuning sample sizes. m is the slope and R^2 measures the linearity of the trend. Bold represents best scaling

Configuration	$N_{top} = 10^4$		$N_{top} = N_{max}$	
	m	R^2	m	R^2
Classifier	-67.0	0.197	432.5	0.994
Generator	-28.7	0.419	59.5	0.353
MPM	83.7	0.969	166.8	0.912
Classifier + Generator	27.5	0.088	178.1	0.960
Classifier + MPM	3.4	0.001	212.7	0.878
Generator + MPM	-48.1	0.673	93.5	0.705
Classifier + Generator + MPM	38.6	0.076	134.6	0.844
Average (over pre-train modes)	1.4	0.346	182.5	0.807

The low- N_{top} regime is quite different. At $N_{top} = 10^4$ (left panel of Fig. 7), the average pre-training-mode slope collapses to $\bar{m} = 1.4$ with $\bar{R}^2 = 0.346$, and several configurations have *negative* slopes. Interestingly, the only configuration that shows strong positive scaling with low downstream data is standalone MPM. These results may imply (depending on how strongly conditioned this result is on the micro model size) that the value of additional pre-training data is highly dependent on the size of the downstream dataset. In the high-finetuning data regime, more pre-training is a near-universal good and supervised classifier pre-training gains the most. In the low-finetuning data regime on the other hand, the choice of objective dominates over the amount of pre-training data, and most configurations show essentially no improvement as N_{pre} grows.

B. JetNet Generation

The performance of the generative models is evaluated both on particle-level and jet-level observables based on generated samples of 50k jets per type. Uncertainties are estimated with bootstrapping by calculating each metric five times, drawing batches of 10k jets from the generated sample and the JetNet test set with replacement. The Wasserstein distance W_1 is used as a metric to describe how well the generated distribution matches the target (=JetNet) distribution. The evaluated particle-level features are the features directly generated by the model, i.e. $\Delta\eta_i$, $\Delta\phi_i$, $\log p_{T,i}$ and $\log E_i$. The jet-level features quantify if inter-particle correlations are modeled accurately. To this end, we calculate the jet mass, the subjettiness [52] ratios $\tau_{21} = \tau_2/\tau_1$ and $\tau_{32} = \tau_3/\tau_2$ as well as the jet pseudorapidity and jet energy from the particle features. The jet energy is the scalar sum of the constituent energy whereas other jet-level features combine multiple particle-level features in a non-trivial way. The jet mass and the subjettiness ratios are calculated from the four-momenta based on $\Delta\phi_i$, $\Delta\eta_i$ and $\log p_{T,i}$ values (assuming massless

TABLE VI. Performance of the different generative models averaged over all jet types. Each model is trained for 520 epochs on the full dataset ($N_{\text{JetNet}} = 550\text{k}$). The shown values represent the Wasserstein distance W_1 between the generated jets and jets from the JetNet dataset.

(a) Medium				
	τ_{21} $\times 10^{-2}$	τ_{32} $\times 10^{-2}$	Jet energy (GeV)	Jet mass (GeV)
From Scratch	6.7 ± 1.1	4.6 ± 1.2	8.8 ± 1.5	1.7 ± 0.4
Generator	1.7 ± 0.3	1.8 ± 0.4	9.5 ± 2.2	1.4 ± 0.3
Classifier + Generator + MPM	1.3 ± 0.2	1.7 ± 0.4	8.8 ± 1.9	1.2 ± 0.2
Classifier + Generator	1.9 ± 0.4	2.1 ± 0.4	9.1 ± 1.1	1.7 ± 0.4
Generator + MPM	1.6 ± 0.2	1.9 ± 0.4	8.8 ± 1.9	1.2 ± 0.3
Classifier	9.1 ± 1.8	6.5 ± 1.6	9.3 ± 2.3	2.9 ± 0.8
MPM	7.9 ± 1.4	5.7 ± 1.5	10.9 ± 2.3	1.8 ± 0.4
Classifier + MPM	9.4 ± 2.0	6.8 ± 1.8	9.8 ± 2.0	3.1 ± 0.9

(b) Small				
	τ_{21} $\times 10^{-2}$	τ_{32} $\times 10^{-2}$	Jet energy (GeV)	Jet mass (GeV)
From Scratch	2.0 ± 0.3	2.1 ± 0.5	8.0 ± 1.2	0.9 ± 0.1
Generator	1.9 ± 0.3	2.0 ± 0.4	9.0 ± 2.3	1.2 ± 0.3
Classifier + Generator + MPM	1.2 ± 0.1	1.4 ± 0.4	8.1 ± 1.4	0.8 ± 0.2
Classifier + Generator	1.1 ± 0.2	1.4 ± 0.3	7.6 ± 1.2	0.7 ± 0.1
Generator + MPM	1.2 ± 0.2	1.4 ± 0.3	8.6 ± 1.3	0.9 ± 0.1
Classifier	3.6 ± 0.5	3.3 ± 0.6	9.2 ± 1.5	1.1 ± 0.2
MPM	3.0 ± 0.4	2.5 ± 0.6	9.2 ± 2.2	1.0 ± 0.2
Classifier + MPM	4.8 ± 0.8	4.0 ± 0.8	8.5 ± 1.4	1.2 ± 0.2

(c) Micro				
	τ_{21} $\times 10^{-2}$	τ_{32} $\times 10^{-2}$	Jet energy (GeV)	Jet mass (GeV)
From Scratch	10.9 ± 2.4	6.7 ± 1.5	10.1 ± 1.6	4.8 ± 1.6
Generator	10.4 ± 3.4	5.5 ± 1.2	11.6 ± 3.1	10.9 ± 3.5
Classifier + Generator + MPM	13.4 ± 3.0	8.8 ± 0.7	12.9 ± 3.2	12.8 ± 2.5
Classifier + Generator	10.4 ± 2.6	7.0 ± 1.2	12.8 ± 1.8	4.9 ± 1.9
Generator + MPM	13.0 ± 3.0	7.9 ± 0.7	11.4 ± 2.7	10.3 ± 2.9
Classifier	15.3 ± 3.1	6.7 ± 1.5	11.4 ± 1.9	8.4 ± 2.3
MPM	13.4 ± 2.6	7.0 ± 1.5	9.9 ± 2.2	7.0 ± 2.2
Classifier + MPM	11.8 ± 2.6	6.1 ± 1.4	11.3 ± 1.8	5.8 ± 2.0

particle). The absolute value of the jet pseudorapidity η_{jet} is obtained by averaging over the per-particle estimate $\eta_{\text{jet},i} \approx -\Delta\eta_i + \cosh^{-1}(E_i/p_{T,i})$. All metrics are first calculated for each jet type individually and then averaged to reflect the overall performance across all five jet types. The performance of the models trained on the whole JetNet dataset is shown in Table VI, where a selection of metrics is presented. Further metrics can be found in the Appendix in Table XX.

a. From scratch baselines The small model size provides the best from scratch baseline across the different model sizes. The performance of the micro models is by a margin lower than the corresponding small or medium counterparts, potentially due to the limited model size that is simply not sufficient for this generative task. The medium model is significantly better than the micro model, but also falls short in performance compared to the from scratch baseline obtained with the small model size.

We only consider particles with $E_i \geq p_{T,i}$ in this average, thus removing unphysical particles. The fraction of unphysical particles with $E_i < p_{T,i}$ is at the order of $\mathcal{O}(1\%)$.

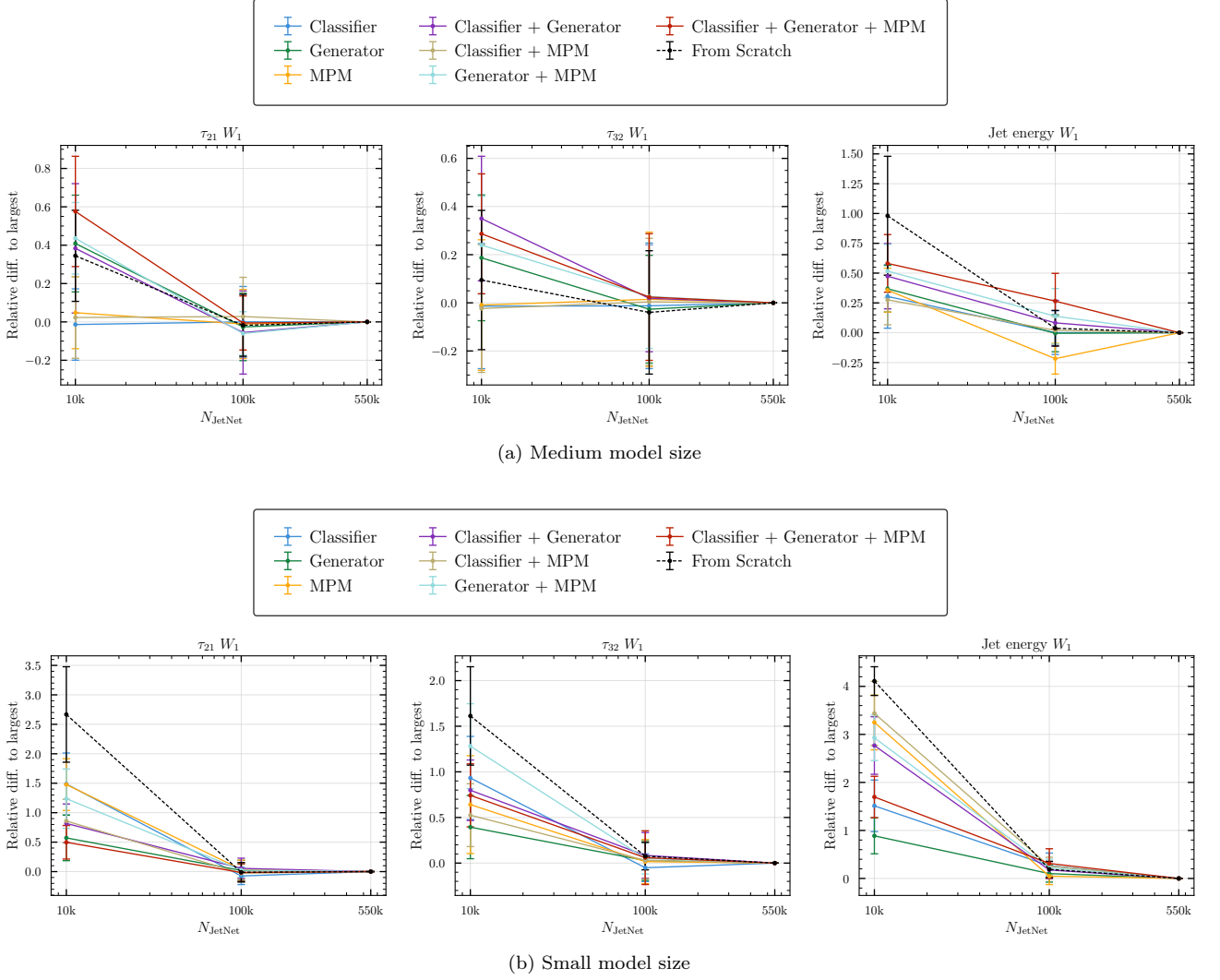


FIG. 8. Relative performance difference of the generative model as a function of the training dataset size compared to the corresponding model obtained with the full dataset size ($N_{\text{JetNet}} = 550\text{k}$) for the medium model (top) and the small model (bottom). The relative performance difference is given by $(W_1(N_{\text{JetNet}}) - W_1(550\text{k})) / W_1(550\text{k})$ for the corresponding metric.

b. Different pre-training strategies Small and medium models show the best performance when pre-trained solely or partially on the generative task. The combination of all three pre-training tasks (Classifier+Generator+MPM) leads to the best downstream generative performance for the medium model while the Classifier+Generator pre-training strategy is the best-performing for the small model. Most notably, all methods that include the generative component during pre-training outperform the from scratch baseline. This is not true for models that did not include the generative task during pre-training. Those models show worse performance than the from scratch baseline, which already indicates that those pre-training strategies are not well-aligned with the generative task. Micro models show worse than from scratch performance on most metrics across all pre-training methods. Yet, the effect seen for small and medium models is inverted here: models pre-trained on the generative task show lower performance than the other pre-training methods. This might be due to the limited model capacity, and the generative model (which was pre-trained to generate JetClass-like jets) not being able to adjust to the new dataset. Due to these limitations, the micro model is omitted in the subsequent evaluation.

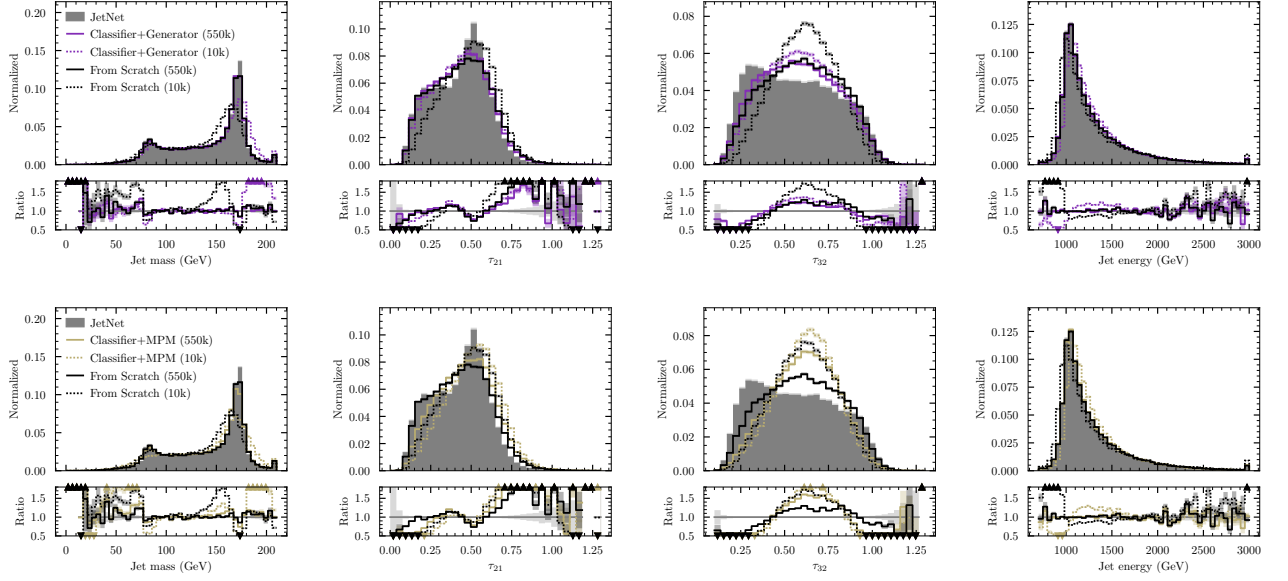


FIG. 9. Jet-level distributions of generated **top jets** obtained from the small model with two different pre-training strategies: Classifier+Generator (top panel) and Classifier+MPM (bottom panel) compared with the from scratch baseline. Two downstream dataset sizes are shown in dotted ($N_{\text{JetNet}} = 10\text{k}$) and solid ($N_{\text{JetNet}} = 550\text{k}$) lines. The gray histogram shows jets from the JetNet dataset.

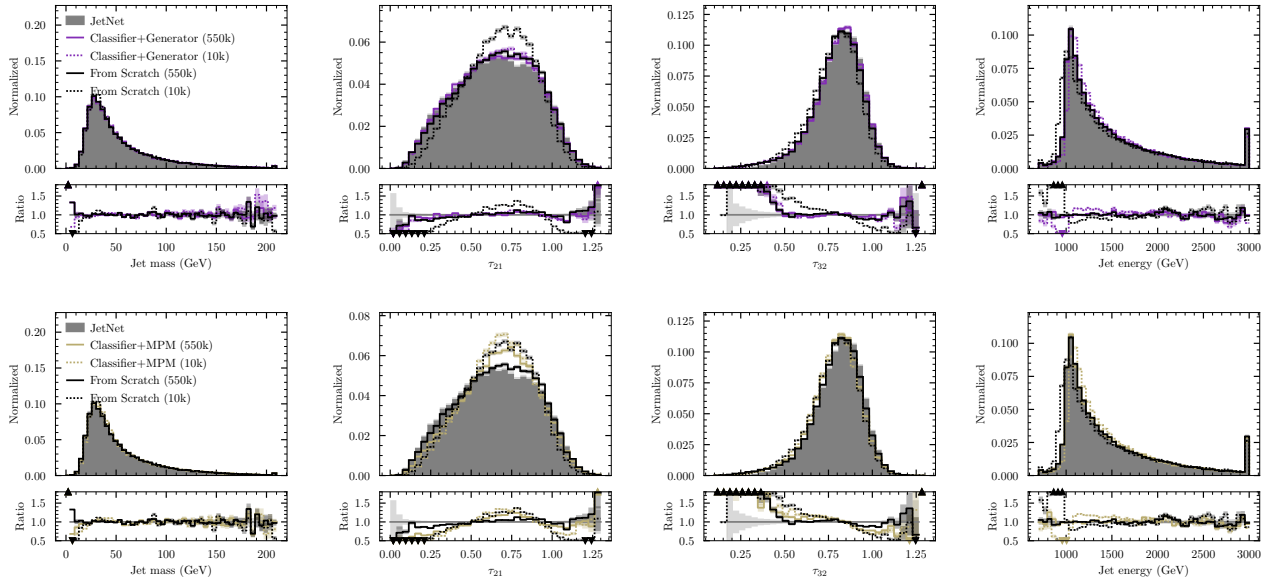


FIG. 10. Jet-level distributions of generated **light quark jets** obtained from the small model with two different pre-training strategies: Classifier+Generator (top panel) and Classifier+MPM (bottom panel) compared with the from scratch baseline. Two downstream dataset sizes are shown in dotted ($N_{\text{JetNet}} = 10\text{k}$) and solid ($N_{\text{JetNet}} = 550\text{k}$) lines. The gray histogram shows jets from the JetNet dataset.

1. Scan over dataset size

To assess the effect of downstream dataset size in the generative task, the performance difference of models trained on either 10k or 100k JetNet jets is compared to the values shown in Table VI. The corresponding relative performance difference among different downstream dataset sizes is shown in Fig. 8, illustrating how close the different models get to the best performance (at $N_{\text{JetNet}} = 550\text{k}$ with the same pre-training strategy). The Figure shows the corresponding

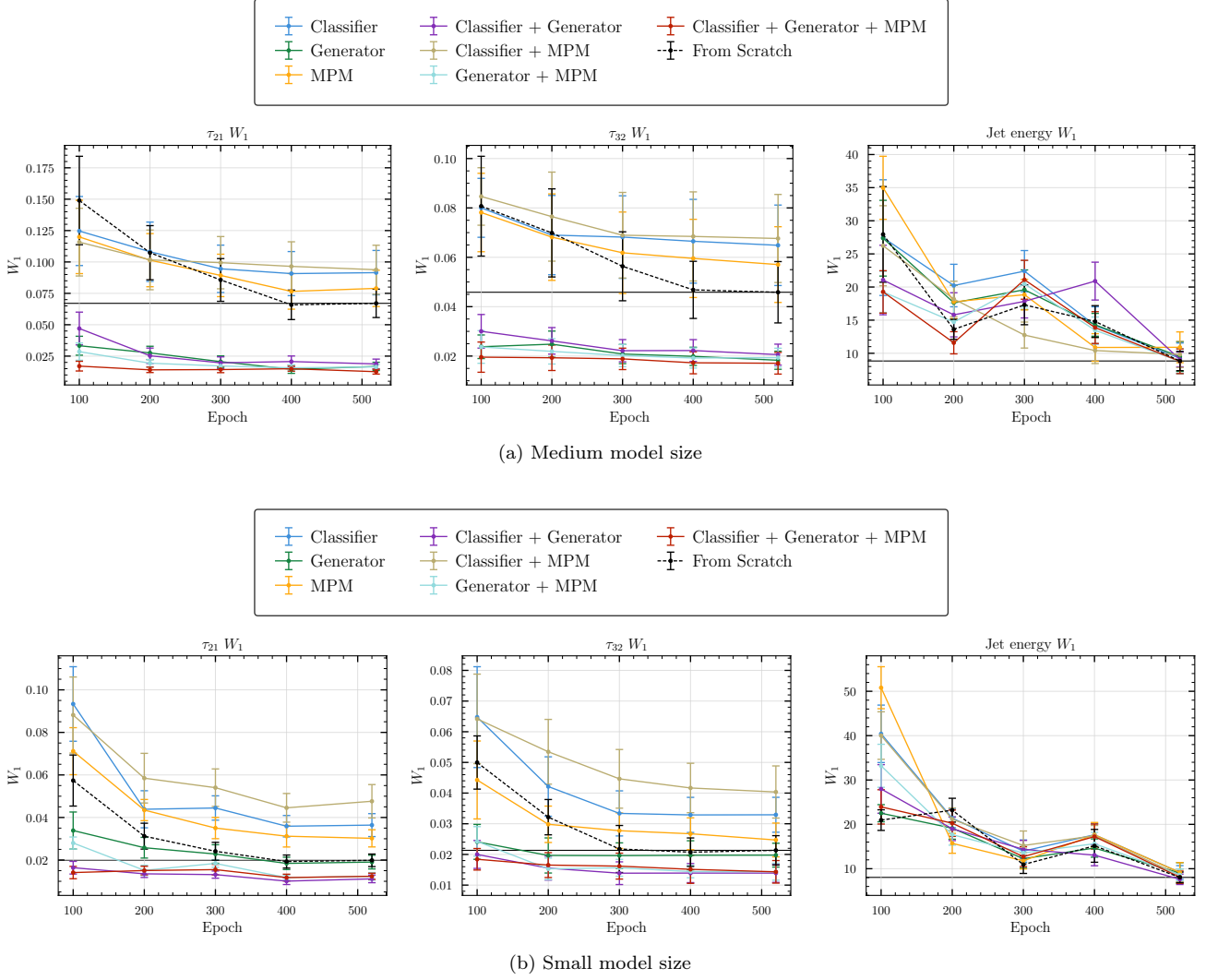


FIG. 11. Performance of the generative models as a function of the training epoch using a training dataset size $N_{\text{JetNet}} = 550\text{k}$ for (a) the medium and (b) the small model size. The black horizontal line marks the value corresponding to the final epoch of the from scratch baseline.

metrics for small and medium model sizes and all seven pre-training methods. All models benefit from being exposed to a larger training sample during downstream generative training. The extent of performance improvement varies for the different metrics and different model sizes. The most notable change is seen when increasing the dataset size from 10k to 100k jets. Only minor improvements are seen when further increasing the dataset size to $N_{\text{JetNet}} = 550\text{k}$.

For medium-sized models with generative pre-training, the Wasserstein distance calculated on the subjettness ratio decreases by around 20% to 60% when going from $N_{\text{JetNet}} = 10\text{k}$ to $N_{\text{JetNet}} = 550\text{k}$. The medium size from scratch baseline also shows improved performance for larger N_{JetNet} , for example with a 100% improvement in the jet energy W_1 when going from $N_{\text{JetNet}} = 10\text{k}$ to $N_{\text{JetNet}} = 550\text{k}$. No clear performance gain is seen when varying the training dataset size for pre-trained models that were not pre-trained on the generative task. This however does not mean that those models perform well for all downstream dataset sizes. Instead, those models show bad agreement with the target distributions no matter how large the training dataset they were exposed to during downstream training on the JetNet dataset.

For small models we observe even larger performance gains with larger N_{JetNet} . A massive improvement is seen in the jet energy modeling, ranging from 100% (Generator-only) to 400% (From Scratch) when comparing the smallest and largest dataset size. In the subjettness ratio Wasserstein distances, the from scratch model shows improvements of 260% in τ_{21} and 160% in τ_{32} , showing that the small from scratch baseline benefits the most from an increase in the

TABLE VII. Improvement $\Delta W_1(f)$ of the Wasserstein distance metric for different features f between epoch 100 and epoch 520.

(a) Medium model size				
	$\Delta W_1(\tau_{21})$ $\times 10^{-2}$	$\Delta W_1(\tau_{32})$ $\times 10^{-2}$	$\Delta W_1(\text{Jet energy})$ (GeV)	$\Delta W_1(\text{Jet mass})$ (GeV)
From Scratch	8.2 ± 3.7	3.5 ± 2.4	19.1 ± 7.4	6.5 ± 1.8
Generator	1.6 ± 0.8	0.5 ± 0.6	17.9 ± 6.1	2.2 ± 1.2
Classifier + Generator + MPM	0.4 ± 0.4	0.2 ± 0.8	10.5 ± 3.7	4.2 ± 1.3
Classifier + Generator	2.8 ± 1.4	0.9 ± 0.8	12.0 ± 5.4	6.2 ± 2.1
Generator + MPM	1.2 ± 0.7	0.4 ± 0.8	10.5 ± 3.6	3.2 ± 1.1
Classifier	3.3 ± 3.3	1.5 ± 2.0	18.2 ± 9.0	8.9 ± 1.4
MPM	4.1 ± 3.3	2.1 ± 2.2	24.1 ± 5.3	5.2 ± 1.8
Classifier + MPM	2.2 ± 3.3	1.7 ± 2.1	16.4 ± 6.4	9.0 ± 2.1

(b) Small model size				
	$\Delta W_1(\tau_{21})$ $\times 10^{-2}$	$\Delta W_1(\tau_{32})$ $\times 10^{-2}$	$\Delta W_1(\text{Jet energy})$ (GeV)	$\Delta W_1(\text{Jet mass})$ (GeV)
From Scratch	3.8 ± 1.2	2.9 ± 1.0	12.9 ± 2.6	2.8 ± 0.4
Generator	1.5 ± 0.9	0.4 ± 0.7	13.5 ± 3.0	1.7 ± 0.9
Classifier + Generator + MPM	0.2 ± 0.3	0.4 ± 0.5	15.8 ± 4.1	0.6 ± 0.3
Classifier + Generator	0.5 ± 0.3	0.6 ± 0.5	20.4 ± 5.6	0.8 ± 0.3
Generator + MPM	1.6 ± 0.3	1.0 ± 0.6	24.6 ± 5.0	1.4 ± 0.4
Classifier	5.7 ± 1.8	3.2 ± 1.7	31.2 ± 6.7	2.8 ± 1.0
MPM	4.1 ± 1.2	2.0 ± 1.4	41.6 ± 5.2	2.3 ± 0.8
Classifier + MPM	4.1 ± 2.0	2.4 ± 1.7	31.5 ± 5.5	2.6 ± 1.0

dataset size. To put these numbers into context, distributions of top jets obtained from the methods with the highest (Classifier+Generator) and lowest (Classifier+MPM) performance on the small model size are shown in Fig. 9 for 10k and 550k training jets, compared to the corresponding from scratch baselines. The from scratch baseline shows a clear improvement with more training data, leading to much better agreement with the target distribution. However, the difference is visually much less dominant for the pre-trained models. While both methods benefit from additional training data during fine-tuning, the overall performance difference between the different pre-training methods is clearly visible. The model pre-trained with the Classifier+Generator method also improves when the dataset size is increased, but the improvements are visually less dominant. Even more important, the Classifier+Generator model fine-tuned with $N_{\text{JetNet}} = 10\text{k}$ jets clearly outperforms the corresponding from scratch baseline. This is not true for the model pre-trained with the Classifier+MPM method, which does not show a benefit over the from scratch model at either of the downstream dataset sizes. It should be noted that top jets are the most challenging jets to model within the JetNet dataset, and were chosen for Fig. 9 as the mentioned trends are most visible there. Other jet types show the same overall trends, with the distribution of the generated jets showing better agreement with the target distribution. This can be seen in Fig. 10 for light quark jets with more jet types found in appendix plots Fig. 20 and Fig. 21.

2. Performance vs. training duration

To investigate how the pre-training affects the convergence speed of the downstream training, we also evaluate the models trained on the whole JetNet dataset as a function of the training epoch. The corresponding curves of the previously introduced metrics are shown in Fig. 11. Further plots with the extended set of metrics are shown in Fig. 22 and Fig. 23 in App. D. Some features, e.g. the jet energy shown in Fig. 11 do not show a clear improvement in convergence speed when pre-trained with either of the investigated methods. Other features such as the subjettness ratios, however, show a clear trend. Those show again the clear overall performance gain obtained from generative pre-training (single-objective or combined with classification or MPM), with those models performing better than the from scratch baseline at any of the evaluated checkpoints. Even more, the generative pre-trained models already outperform the final from-scratch performance after 100 training epochs. For small models this effect is less pronounced, but still visible. Generative pre-trained models with that model size surpass the final from scratch performance in τ_{21} and

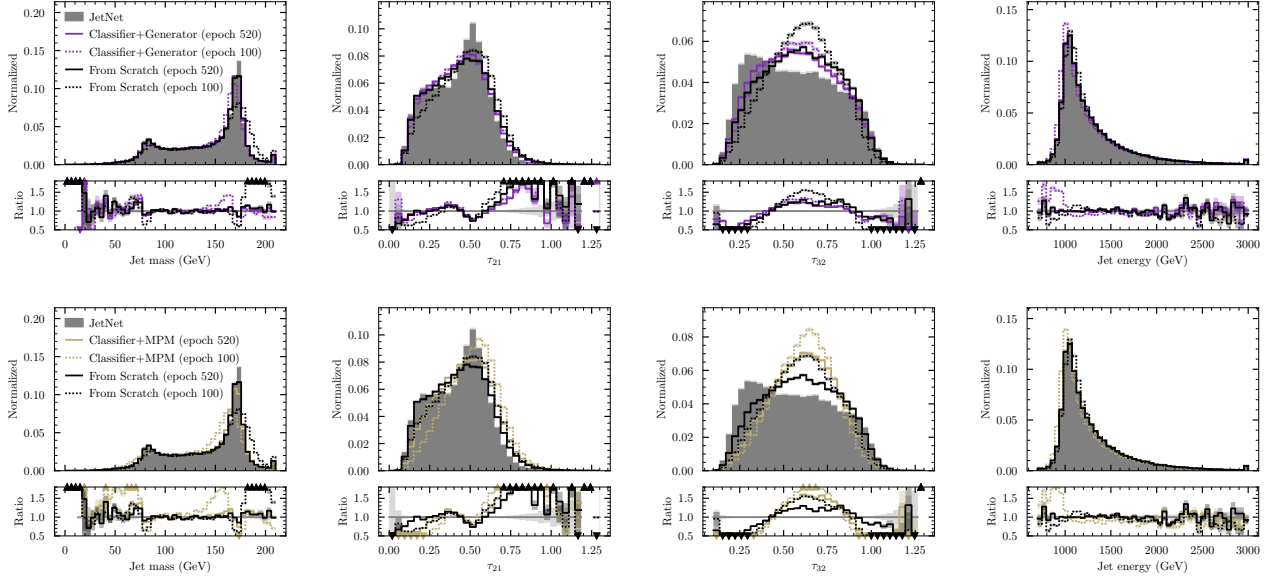


FIG. 12. Jet-level distributions of generated top jets obtained from the small model with two different pre-training strategies: Classifier+Generator (top panel) and Classifier+MPM (bottom panel) compared with the from scratch baseline. Two evaluated epochs are shown in dotted (epoch 100) and solid (epoch 520) lines. The gray histogram shows jets from the JetNet dataset.

τ_{32} already after 100 or 200 epochs, except for the generator-only pre-trained model that takes 400 epochs to surpass the final from-scratch performance. Another notable effect seen in the subjeettiness ratios is the overall improvement from epoch 100 to epoch 520, with the corresponding values shown in Table VII. Most generative pre-trained models only show modest improvements between epoch 100 and epoch 520, indicating that even short fine-tunings of those models can yield performance close to the performance obtained with longer downstream training. The remaining pre-training methods, together with the from scratch baseline, show larger performance gains when trained for longer, revealing that the corresponding pre-trainings yield a model state that needs significantly more computationally expensive fine-tuning. The respective distributions of top jets are shown in Fig. 12 for the Classifier+Generator, the Classifier+MPM and the from scratch model, where this effect is also visually noticeable, underlining yet again the importance of including the generative task during pre-training if downstream generation is of interest. As mentioned earlier in the comparison of different downstream dataset sizes, the overall quality of the generated jets is much better for the other jet types, with the corresponding plots shown in the appendix in Fig. 24 and Fig. 25.

V. CONCLUSION

We present a study of pre-training objectives for jet foundation models, using the OMNILEARNED framework. Holding the architecture fixed and varying only the active task heads, we benchmark seven pre-training configurations across three axes, pre-training dataset size, finetuning dataset size, and model size. All models are pre-trained on JetClass and fine-tuned on two representative downstream tasks, top-quark jet tagging, and JetNet conditional generation. We aim to disentangle the contributions of supervised, generative, and self-supervised signals to the downstream transfer, and to map out how that picture depends on the amount of data and the capacity available at both stages, given we have access to labels at scale.

The top-tagging results show that no single pre-training objective is universally optimal. When labeled datasets for pre-training and model capacity are available, plain supervised classifier pre-training is the best configuration at every model size and scales near-linearly with the amount of pre-training data when performing downstream classification. The ranking inverts in the low-label regime, at 10^4 top labels, Classifier+MPM is the top-ranked configuration at every model size, and at the smallest model size, standalone MPM outperforms pure classifier pre-training despite using no labels at all. Standalone MPM is also the most scale-efficient single-objective configuration with respect to model capacity, with a slope roughly $2\times$ that of the pure classifier even though the classifier pre-training is still the most performant. We further found that pre-training loss does not reliably track downstream performance. Instead, downstream performance peaks well before pre-training converges. The value of additional pre-training data may depend sharply on the downstream sample size, ranging from near-linear gains when downstream labels are abundant

to essentially no gain (or, for the pure classifier, mild degradation) when downstream labels are scarce. This effect needs to be confirmed at larger model sizes. Finally, we notice that standalone flow-matching-based generative pre-training does not lead to a performance improvement in downstream classification. This indicates that the flow matching task is not well aligned with the classification task, and that the learned representation can not be easily transferred to the classification task. Yet, the classification performance can be enhanced by combining flow matching with either MPM in a self-supervised setting or with classification if labels are available during pre-training. This result, i.e. that purely generative pre-training can yield sub-optimal pre-trained model components, but can be improved in a self-supervised way by being combined with MPM is similar to the findings in [6], where this combination was studied in the form of next token prediction (generative) and masked token prediction (MPM).

Our studies on the generative downstream task based on the JetNet dataset show the importance of aligning the pre-training task(s) with the downstream task. While the performance obtained with different pre-training strategies varies among the different feature distributions investigated here, we notice that neither the Classifier nor the MPM task help for downstream generation quality, unless combined with the generative task during pre-training. All models that were pre-trained at least partially on the generative task show a performance gain when fine-tuned to jet generation on another dataset. Not only do these fine-tuned models outperform the corresponding from-scratch baseline, but the gap between the performance early on and at the end of the downstream training is much smaller for jet substructure features like the subjettness ratios, often surpassing the results obtained from the model trained from scratch with 5 times fewer training epochs.

Taken together, these results suggest that the optimal pre-training objective is not a property of the model alone but of the joint configuration of model capacity, pre-training data, and downstream data availability and compatibility. The combination of multiple objectives, while not necessarily always the best performing for each individual task, provides a balance in benefits from both downstream tasks studied in this work. Additional pre-training targets avoid over-specialization from the supervised pre-training and promote robustness and improved transferability during downstream tasks.

Three directions for future work follow naturally from this study. First, our finetuning-trajectory results show that pre-training loss is an unreliable proxy for downstream performance, with downstream background rejection peaking well before pre-training converges. Understanding this effect, its scaling, and mitigating it by introducing pre-training modifications (including multi-objective pre-training) would be an interesting next step. It may be necessary to add additional downstream tasks, especially those with less overlap with the pre-training tasks.

Second, we have benchmarked seven configurations at three model sizes spanning $\sim 100\text{k}$ to $\sim 51\text{M}$ parameters, which we believe is sufficient to reveal pre-training ordering, but does not yet probe the regime where the largest HEP foundation models operate ($\sim 10^9$ parameters and pre-training on billions of tokens [1]). Three configurations are of particular interest at this scale, pure Classifier, which dominates when capacity and labels are plentiful and seem to have the cleanest data scaling, standalone MPM, which exhibits the steepest parameter-scaling and data scaling (at low downstream dataset sizes) slope among single-objective configurations despite lower absolute performance (which may change with scale), and Classifier+Generator+MPM, which combines the strengths of all three signals. Scaling these three configurations to the size of the published OMNILEARNED-I model and to the full $\sim 10^9$ -jet pre-training would reveal the continuation of these trends.

Finally, understanding the origin of the seeming orthogonality between generative and classification tasks would be quite interesting. Classification focuses on the differences between datasets while generation focuses on the bulk of the datasets - while a shared latent space may be effective at solving both types of problems, it need not be. The difference in model representation between the two pre-training tasks may be a good probe to answer this and broader interpretability questions.

CODE AVAILABILITY

The code used for this study can be found at <https://github.com/ibrahimEls/PretrainingForScience>.

ACKNOWLEDGMENTS

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award HEP-ERCAP0035546. I.E. is supported in part by the Connaught International Scholarship at University of Toronto and Natural Sciences and Engineering Research Council of Canada (NSERC) Canada Graduate Research Scholarship. I.E. thanks Yonatan Kahn, Andrew Larkoski, and David Curtin for valuable conversations on various aspects of parton showers. I.E. thanks Marat Freytsis and Anthropic for access to Claude

Code, which assisted with aspects of orchestrating training models used in this work. V.M. is supported by JST EXPERT-J, Japan Grant Number JPMJEX2509. J.B. and G.K. are supported by the DFG under the German Excellence Initiative – EXC 2121 Quantum Universe – 390833306. J.B. is supported by a scholarship of the German Academic Exchange Service (DAAD). J.B. also acknowledges support via the Hamburg VISTA/VISOR — Virtual Initiative for Science & Technology in AI — network. B.N. is supported by the Department of Energy (DOE), Office of Science under contract DE-AC02-76SF00515.

-
- [1] W. Bhimji, C. Harris, V. Mikuni, and B. Nachman, *Phys. Rev. D* **113**, 032020 (2026), [arXiv:2510.24066 \[hep-ph\]](#).
 - [2] V. Mikuni and B. Nachman, *Phys. Rev. D* **111**, L051504 (2025), [arXiv:2404.16091 \[hep-ph\]](#).
 - [3] V. Mikuni and B. Nachman, *Phys. Rev. D* **111**, 054015 (2025), [arXiv:2502.14652 \[hep-ph\]](#).
 - [4] J. Birk, A. Hallin, and G. Kasieczka, *Mach. Learn. Sci. Tech.* **5**, 035031 (2024), [arXiv:2403.05618 \[hep-ph\]](#).
 - [5] O. Amram, L. Anzalone, J. Birk, D. A. Faroughy, A. Hallin, G. Kasieczka, M. Krämer, I. Pang, H. Reyes-Gonzalez, and D. Shih, *Mach. Learn. Sci. Tech.* **6**, 030601 (2025), [arXiv:2412.10504 \[hep-ph\]](#).
 - [6] J. Birk, A. Hallin, G. Kasieczka, N. Madzharova, I. Pang, and D. Shih, *Mach. Learn. Sci. Tech.* **7**, 035042 (2026), [arXiv:2512.04149 \[hep-ph\]](#).
 - [7] M. Vigl, N. Hartman, and L. Heinrich, *Mach. Learn. Sci. Tech.* **5**, 025075 (2024), [arXiv:2401.13536 \[hep-ex\]](#).
 - [8] L. Tani, J. Pata, and J. Birk, *SciPost Phys. Core* **8**, 046 (2025), [arXiv:2503.19165 \[hep-ex\]](#).
 - [9] M. McCabe, P. Mukhopadhyay, T. Marwah, B. R.-S. Blancard, F. Rozet, C. Diaconu, L. Meyer, K. W. K. Wong, H. Soutoudeh, A. Bietti, I. Espejo, R. Fear, S. Golkar, T. Hehir, K. Hirashima, G. Krawezik, F. Lanusse, R. Morel, R. Ohana, L. Parker, M. Pettee, J. Shen, K. Cho, M. Cranmer, and S. Ho, *Walrus: A cross-domain foundation model for continuum dynamics* (2025), [arXiv:2511.15684 \[cs.LG\]](#).
 - [10] L. Parker, F. Lanusse, S. Golkar, L. Sarra, M. Cranmer, A. Bietti, M. Eickenberg, G. Krawezik, M. McCabe, R. Morel, R. Ohana, M. Pettee, B. Régalo-Saint Blancard, K. Cho, and S. a. Ho, *Monthly Notices of the Royal Astronomical Society* **531**, 4990–5011 (2024).
 - [11] A. Morehead, M. Cretu, A. Panescu, R. Anand, M. Weiler, T. Perez, S. Blau, S. Farrell, W. Bhimji, A. Jain, H. Sasrabuddhe, P. Lio, T. Jaakkola, R. Gomez-Bombarelli, R. Ying, N. B. Erichson, and M. W. Mahoney, *Zatom-1: Towards a multimodal foundation model for 3d molecules and materials* (2026), [arXiv:2602.22251 \[cs.LG\]](#).
 - [12] B. M. Wood, M. Dzamba, X. Fu, M. Gao, M. Shuaibi, L. Barroso-Luque, K. Abdelmaqsoud, V. Gharakhanyan, J. R. Kitchen, D. S. Levine, K. Michel, A. Sriram, T. Cohen, A. Das, A. Rizvi, S. J. Sahoo, Z. W. Ulissi, and C. L. Zitnick, *Uma: A family of universal models for atoms* (2026), [arXiv:2506.23971 \[cs.LG\]](#).
 - [13] P. Harris, J. Krupa, M. Kagan, B. Maier, and N. Woodward, *Phys. Rev. D* **111**, 032010 (2025), [arXiv:2403.07066 \[hep-ph\]](#).
 - [14] T. Golling, L. Heinrich, M. Kagan, S. Klein, M. Leigh, M. Osadchy, and J. A. Raine, *Mach. Learn. Sci. Tech.* **5**, 035074 (2024), [arXiv:2401.13537 \[hep-ph\]](#).
 - [15] M. Leigh, S. Klein, F. Charton, T. Golling, L. Heinrich, M. Kagan, I. Ochoa, and M. Osadchy, *Machine Learning: Science and Technology* **6**, 025075 (2025), [arXiv:2409.12589 \[hep-ph\]](#).
 - [16] J. Bardhan, R. Agrawal, A. Tilak, C. Neeraj, and S. Mitra, *HEP-JEPA: A foundation model for collider physics using joint embedding predictive architecture* (2025), [arXiv:2502.03933 \[cs.LG\]](#).
 - [17] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, *Comput. Phys. Commun.* **191**, 159 (2015), [arXiv:1410.3012 \[hep-ph\]](#).
 - [18] J. Bellm *et al.*, *Eur. Phys. J. C* **76**, 196 (2016), [arXiv:1512.01178 \[hep-ph\]](#).
 - [19] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi (DELPHES 3), *JHEP* **02**, 057, [arXiv:1307.6346 \[hep-ex\]](#).
 - [20] G. Aad *et al.* (ATLAS), *Phys. Rev. Lett.* **124**, 222002 (2020), [arXiv:2004.03540 \[hep-ex\]](#).
 - [21] S. Agostinelli *et al.* (GEANT4), *Nucl. Instrum. Meth. A* **506**, 250 (2003).
 - [22] H. Qu, C. Li, and S. Qian, in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, Vol. 162 (PMLR, 2022) pp. 18281–18292, [arXiv:2202.03772 \[hep-ph\]](#).
 - [23] J. Batson and Y. F. Kahn, *SciPost Phys. Core* **8**, 034 (2025), [arXiv:2312.02264 \[hep-ph\]](#).
 - [24] ATLAS Collaboration, *Carpe Datum: Scaling behavior of transformers for heavy hadron flavor identification*, Tech. Rep. ATL-SOFT-PUB-2026-002 (CERN, 2026).
 - [25] M. Vigl, N. Hartman, M. Kagan, and L. Heinrich, *Neural Scaling Laws for Boosted Jet Tagging* (2026), [arXiv:2602.15781 \[hep-ex\]](#).
 - [26] O. Amram, D. A. Faroughy, T. Gerdes, A. Hallin, G. Kasieczka, M. Krämer, H. Reyes-Gonzalez, and D. Shih, *Neural Scaling Laws for Jet Generation* (2026), [arXiv:2605.28940 \[hep-ph\]](#).
 - [27] H. Bahl, V. Bresó-Pla, A. Butter, and J. I. Ramirez, *Scaling laws for amplitude surrogates* (2026), [arXiv:2601.13308 \[hep-ph\]](#).
 - [28] S. Mondal and L. Mastrolorenzo, *The European Physical Journal Special Topics* **233**, 2657 (2024).
 - [29] E. Buhmann, C. Ewen, G. Kasieczka, V. Mikuni, B. Nachman, and D. Shih, *Phys. Rev. D* **109**, 055015 (2024), [arXiv:2310.06897 \[hep-ph\]](#).
 - [30] V. Mikuni and B. Nachman, *Searching for Anomalies with Foundation Models* (2026), [arXiv:2603.23593 \[hep-ex\]](#).

- [31] Z. Bogorad, I. Elsharkawy, Y. Kahn, A. J. Larkoski, and N. Levi, Generative models on phase space (2026), [arXiv:2604.02415 \[hep-ph\]](#).
- [32] T. Fauceit, J. Thaler, and D. Whiteson, *Phys. Rev. D* **103**, 036020 (2021), [arXiv:2010.11998 \[hep-ph\]](#).
- [33] I. Elsharkawy, V. Mikuni, W. Bhimji, and B. Nachman, OmniMol: Transferring Particle Physics Knowledge to Molecular Dynamics with Point-Edge Transformers (2026), [arXiv:2601.10791 \[physics.chem-ph\]](#).
- [34] V. Mikuni, I. Elsharkawy, and B. Nachman, OmniCosmos: Transferring Particle Physics Knowledge Across the Cosmos (2025), [arXiv:2512.24422 \[astro-ph.CO\]](#).
- [35] G. Krzmar, V. Mikuni, B. Nachman, and C. Wilkinson, Cross-Domain Transfer with Particle Physics Foundation Models: From Jets to Neutrino Interactions (2026), [arXiv:2604.12364 \[hep-ex\]](#).
- [36] C. Li *et al.*, Accelerating Resonance Searches via Signature-Oriented Pre-training (2024), [arXiv:2405.12972 \[hep-ph\]](#).
- [37] G. Aad *et al.* (ATLAS), *JINST* **19** (08), P08018, [arXiv:2407.20127 \[hep-ex\]](#).
- [38] V. Andreev *et al.* (H1), *Phys. Rev. Lett.* **128**, 132002 (2022), [arXiv:2108.12376 \[hep-ex\]](#).
- [39] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, Flow matching for generative modeling (2023), [arXiv:2210.02747 \[cs.LG\]](#).
- [40] H. Qu, C. Li, and S. Qian, [10.5281/zenodo.6619768](#) (2022).
- [41] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, *JHEP* **07**, 079, [arXiv:1405.0301 \[hep-ph\]](#).
- [42] NERSC, Perlmutter, <https://docs.nersc.gov/systems/perlmutter/architecture/>, HPE Cray EX supercomputer, National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory. Accessed: 2026-05-18.
- [43] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton, in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019).
- [44] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, in *International Conference on Learning Representations* (2020).
- [45] L. Wright, Ranger: a synergistic optimizer combining RAdam (rectified adam) and LookAhead, <https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer> (2019).
- [46] T. Buss, [FLC-QU-hep/ranger-lite: RangerLite 1.0.0](#) (2025).
- [47] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le, in *Advances in Neural Information Processing Systems (NeurIPS)* (2023) [arXiv:2302.06675 \[cs.LG\]](#).
- [48] A. Butter *et al.*, *SciPost Phys.* **7**, 014 (2019), [arXiv:1902.09914 \[hep-ph\]](#).
- [49] G. Kasieczka, T. Plehn, J. Thompson, and M. Russell, [10.5281/zenodo.2603256](#) (2019).
- [50] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant, and D. Gunopulos, in *35th Conference on Neural Information Processing Systems* (2021) [arXiv:2106.11535 \[cs.LG\]](#).
- [51] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant, and D. Gunopulos, [10.5281/zenodo.6975118](#) (2022).
- [52] J. Thaler and K. Van Tilburg, *JHEP* **03**, 015, [arXiv:1011.2268 \[hep-ph\]](#).
- [53] F. A. Dreyer, G. P. Salam, and G. Soyez, *JHEP* **12**, 064, [arXiv:1807.04758 \[hep-ph\]](#).
- [54] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski, in *The Twelfth International Conference on Learning Representations* (2024).

Appendix A: OMNILEARNED Modules

Below is the description of the defining aspects of the OMNILEARNED foundation model. A more detailed description of the model can be found in the original OMNILEARNED and OMNILEARN papers. [1–3]

a. Local Embedding A local block captures short-range correlations between nearby particles before the main body. For each particle i , we identify its $k = 10$ nearest neighbors in (η, ϕ) space and compute, for each pair (x_i, x_j) , the pairwise features

$$f(x_i, x_j) = [x_i - x_j, \log m(x_i, x_j), \log \Delta_R(x_i, x_j), \log k_T(x_i, x_j)], \quad (\text{A1})$$

where $m(x_i, x_j)$ is the invariant mass of the pairwise four-vector sum, Δ_R is the distance in rapidity-azimuth space, and k_T is the pairwise transverse momentum standard in jet analysis [53]. The features $f(x_i, x_j)$ are passed through an MLP and then a transformer block that attends only to the neighbors of each particle. The resulting local representation is added to the per-particle embedding h_i . The total embedding h_i is given by,

$$h_i = \text{MLP}_{\text{kin}}(x_i) + \text{EMB}_{\text{PID}}(\text{PID}) + \text{MLP}_{\text{add}}(v_i) + \text{ATTN}_{\text{local}}(f(x_i, x_j)), \quad x_i = (\Delta\eta_i, \Delta\phi_i, \log p_{T,i}, \log E_i). \quad (\text{A2})$$

where x_i is the input jet constituent feature vector, of said jet constituent, and v_i is additional vertex information for that constituent.

b. Physics-informed attention bias The same pairwise features $f(x_i, x_j)$ from Eq. A1, now evaluated over *all* particle pairs rather than only k -neighbors, are reused in the main body of the model. Each pair is passed through a shared MLP that outputs one scalar per attention head B_{ij} . These scalars are added as an additive bias to the corresponding self-attention logits in every body transformer block as originally introduced in [22].

c. Global attention The main body consists of N_{body} transformer blocks operating on the full set of particle embeddings, with the interaction bias added to every attention matrix. The model carries $N_{\text{tok}} = 4$ learnable global tokens [54] that are concatenated to the particle set from the very first embedding block, so that they attend to all particles throughout the body and aggregate jet-level information.

Appendix B: Hyperparameters

This appendix describes hyperparameters not stated in the main text. Model architecture (depth, embedding width, attention-head count, parameter count, and per-node local batch size) is given in Tab. II.

1. Pre-training Hyperparameters

Tab. VIII lists the learning rate and weight decay used for each size, optimizer, and pair. As described in the main text, configurations whose loss function includes the regression MPM head, as well as all Medium runs, are trained with Ranger. The remaining configurations use Lion. Settings that are common to all pre-training run are listed in Tab. IX.

TABLE VIII. Size and optimizer specific pre-training hyperparameters. Lion is used unless the configuration is Medium or contains an active MPM head, in which case Ranger is used.

	Lion LR / WD	Ranger LR / WD
Micro	$1 \cdot 10^{-3} / 0.01$	$1 \cdot 10^{-3} / 0.01$
Small	$1 \cdot 10^{-4} / 0.30$	$1 \cdot 10^{-3} / 0.01$
Medium	N/A	$5 \cdot 10^{-4} / 0.01$

TABLE IX. Pre-training hyperparameters held fixed across every run. The optimizer hyperparameters listed apply to whichever optimizer is in use for a given configuration Lion or Ranger.

Hyperparameter	Value
Global batch size	8192
Learning-rate schedule	Cosine with linear warmup
Warmup steps	1000
Lion betas (β_1, β_2)	(0.95, 0.99)
Ranger betas (β_1, β_2)	(0.95, 0.999)
Feature dropout	0.1
Gradient accumulation	2
Number of summary tokens	4
Precision	FP32

2. Top Tagging Hyperparameters

All top-tagging finetuning use Lion with the betas of Tab. IX, a cosine learning-rate schedule with linear warmup, a per-GPU batch size of 64, 4 nodes (global batch size 1024), and a fixed number of steps corresponding to a model-dependent epoch budget of 1.2 million top-tagging jets. When a pre-trained body is loaded the classification head receives a $5\times$ multiplier on the base learning rate. Per-size values are summarized in Tab. X and fixed settings are in Tab. XI.

3. Generation Hyperparameters

Hyperparameters for the JetNet generative downstream task are given in Tab. XII. Pre-trained weights are fine-tuned with a learning rate five times smaller than the randomly initialized weights. The maximum training duration

TABLE X. Top-tagging finetuning hyperparameters by model size. “fs” rows use a randomly initialized body.

	Learning rate	Weight decay	Warmup steps	Epoch budget
Micro	$5 \cdot 10^{-5}$	0.1	1028	30
Small	$5 \cdot 10^{-6}$	0.1	1028	30
Medium	$1 \cdot 10^{-6}$	5.0	0	30
Micro fs	$1 \cdot 10^{-4}$	0.5	0	30
Small fs	$5 \cdot 10^{-4}$	0.5	0	15
Medium fs	$1 \cdot 10^{-5}$	0.5	0	30

TABLE XI. Top-tagging hyperparameters held fixed across all sizes.

Hyperparameter	Value
Optimizer	Lion
Optimizer (β_1, β_2)	(0.95, 0.99)
Per-GPU batch size	64
Number of nodes	4
Global batch size	1024
Gradient accumulation	1
Head LR multiplier	$5 \times$ (pre-trained); $1 \times$ (fs)
Learning-rate schedule	Cosine with linear warmup
Precision	FP32

of 70k steps is kept fixed for all model sizes. This leads to a sufficient training duration for the small models, though initial studies showed that medium-sized models can benefit from longer trainings. However, due to computational constraints, the same maximum training step limit is kept for the results presented for the medium size.

TABLE XII. Hyperparameters used for the generative downstream task. The stated learning rate corresponds to the learning rate used for untrained parameters. For pre-trained parameters a factor five smaller learning rate is used.

Hyperparameter	Value
Optimizer	Lion
Optimizer (β_1, β_2)	(0.95, 0.99)
Weight decay	0.01
Global batch size	4096
Max. training steps	70 000
Learning rate	(micro) $2 \cdot 10^{-4}$ (small) $1 \cdot 10^{-4}$ (medium) $5 \cdot 10^{-6}$
Learning-rate schedule	Cosine with linear warmup
Precision	FP32

Appendix C: More Top Jet Classification Results

This section shows similar plots and tables as Section IV A but with the test loss (binary cross-entropy) metric as a function of dataset/pre-training mode/model size/pre-training size. Figures 15 a-c show the loss scaling with respect to top-tagging size for the various pre-training modes for all three model sizes. Figure 16 depicts the test-loss scaling with respect to model size. The linear fits for these scaling are found in Table XIII. Figure 13 and Figure 14 shows the delta performance for all N_{ft} gained by scaling the model from micro to medium. Figure 17 shows test loss scaling with respect to pre-training size. Table XIV gives the scaling with respect to pre-training size. Finally, Table XVIII and Table XIX contain raw pre-training loss and finetuning loss along the trajectory corresponding to Figure 5.

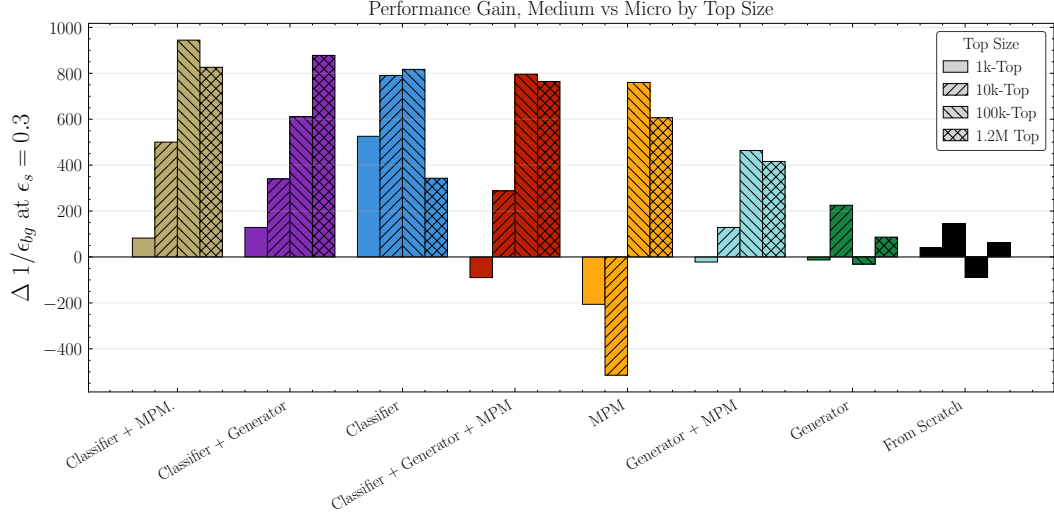


FIG. 13. Top-Tagging $1/\epsilon_{bg}$ performance gain from the micro model to the medium model for all N_{ft} .

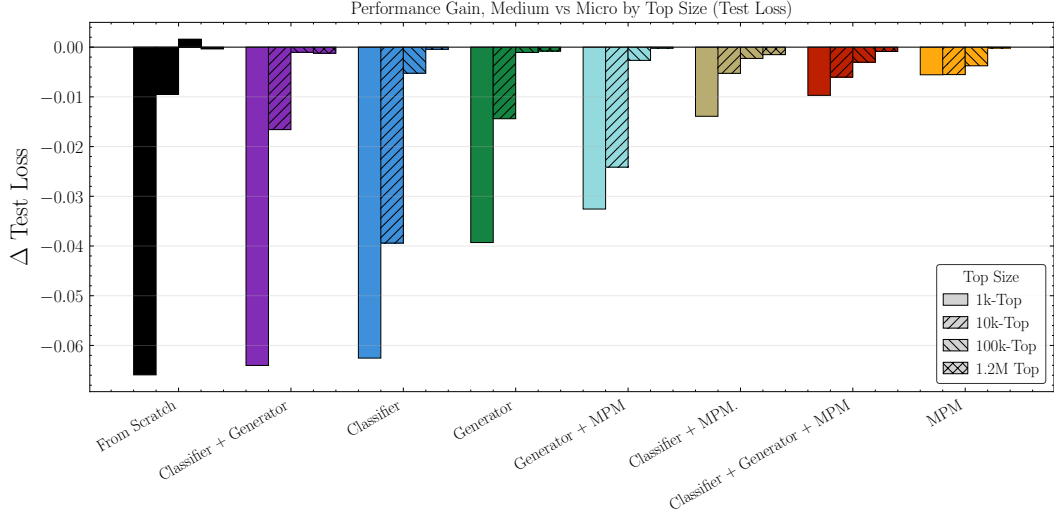


FIG. 14. Top-Tagging Loss performance gain from the micro model to the medium model for all N_{ft}

Appendix D: More Jet Generation Results

This section provides additional results related to the generative downstream task. Table XX shows additional metrics calculated for the evaluation of the generative models. Those metrics complement the ones shown in Table VI in Section IV B, but were found to be less informative in terms of effects that different pre-training methods have on the downstream performance. The corresponding performance difference seen between different dataset sizes is shown for the extended set of metrics in Figures 18 and 19, and the corresponding performance vs. training epoch trajectories are shown in Figures 22 and 23.

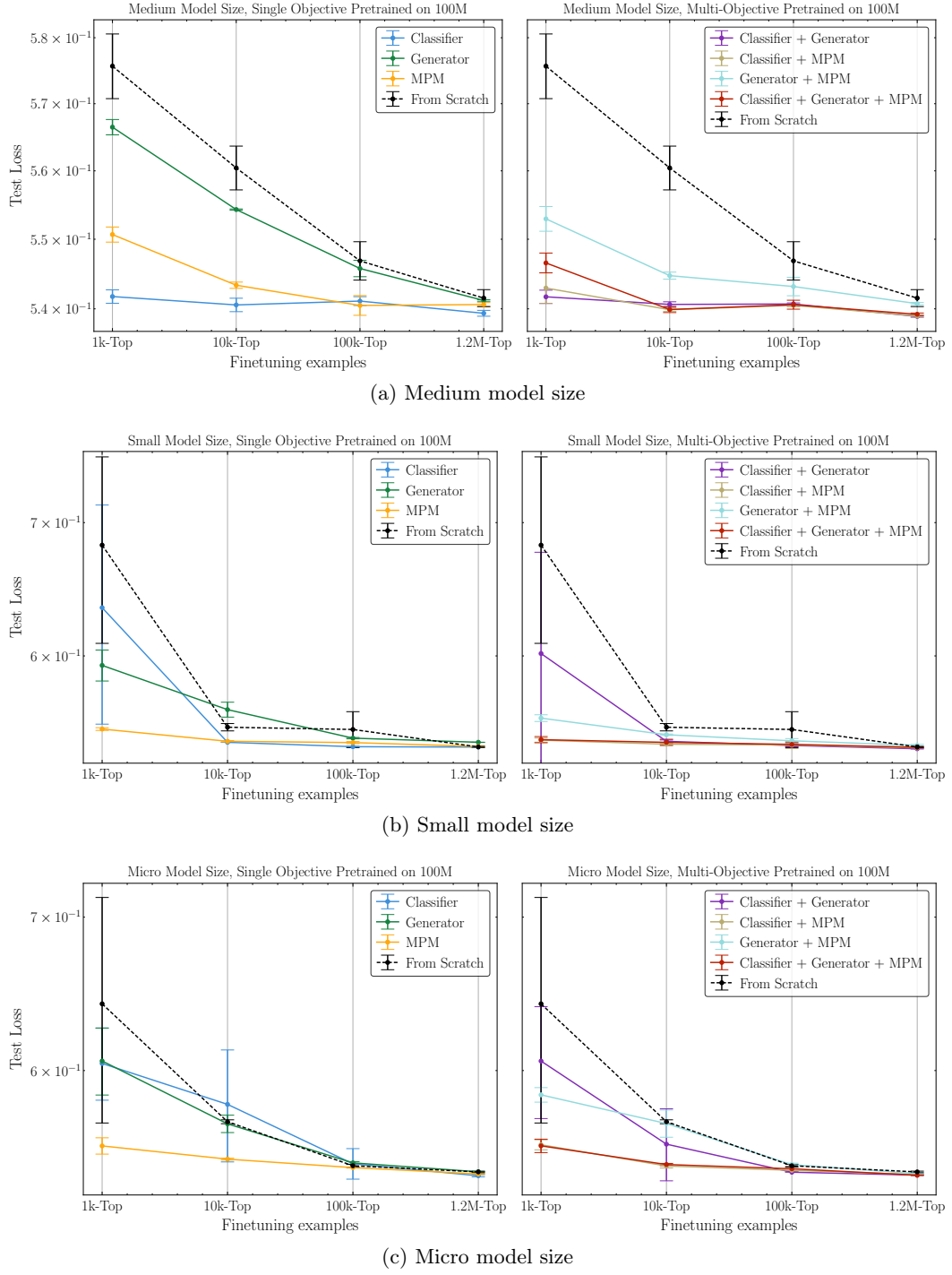


FIG. 15. Loss vs top-tagging dataset size for different model sizes, pre-trained on 100M JetClass jets.

TABLE XIII. Linear fits of test loss vs $\log(N_{\text{params}})$. Slope m gives the loss reduction per decade of parameters, R^2 measures the linearity of the trend. Bold represents the most negative slope (best scaling).

Configuration	$m, (10^{-4})$	R^2
Classifier	-1.20	0.294
Generator	-1.56	0.059
MPM	-0.94	0.345
From Scratch	-2.69	0.146
Classifier + Generator	-4.86	0.899
Classifier + MPM	-5.58	1.000
Generator + MPM	-0.68	0.386
Classifier + Generator + MPM	-2.94	0.837

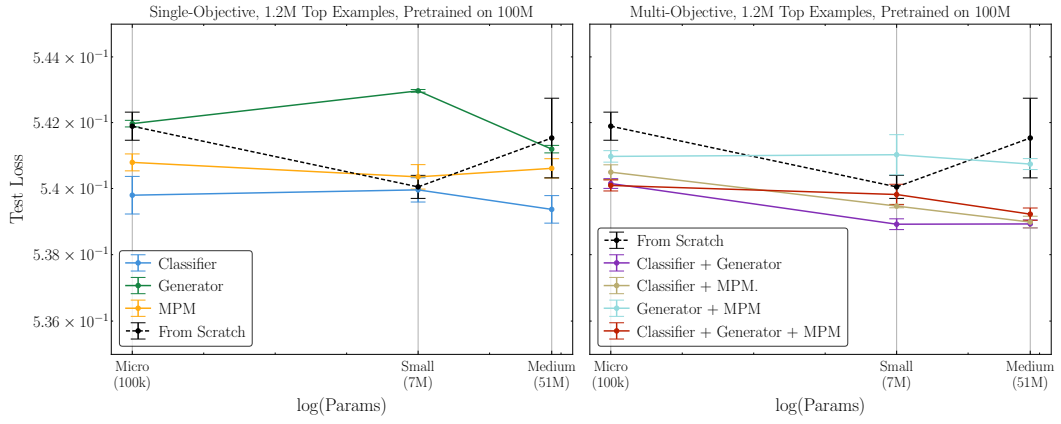


FIG. 16. Top-tagging test loss vs model size for the seven pre-training configurations at $N_{\text{ft}} = N_{\text{max}}$.

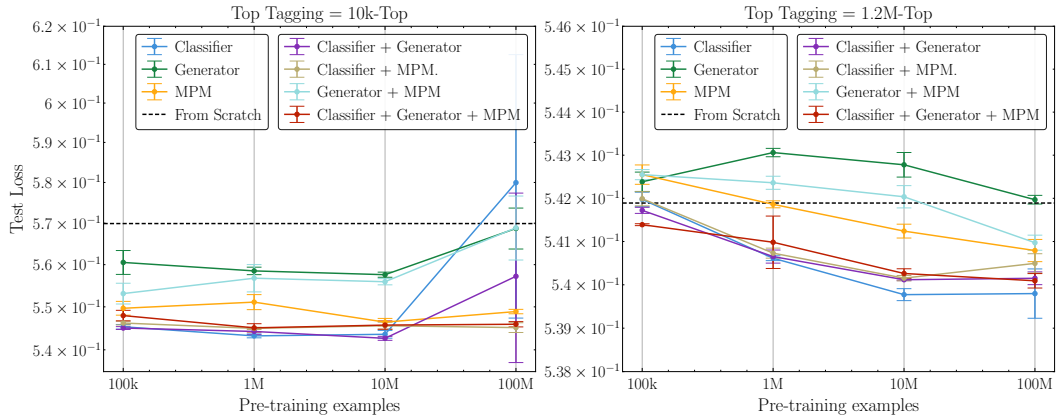


FIG. 17. Loss vs pre-training dataset size N_{pre} for the Micro model under all seven pre-training configurations. $N_{\text{ft}} = 10^4$ is shown on the left. $N_{\text{ft}} = N_{\text{max}} = 1.2 \times 10^6$ is shown on the right. The from-scratch baseline (dashed) does not depend on N_{pre} .

TABLE XIV. Linear fits of validation loss vs $\log(N_{\text{pre}})$ for the Micro model, at two fine-tuning sample sizes. m is the slope (in units of 10^{-3}) and R^2 measures the linearity of the trend. Bold represents best scaling (most negative slope).

Configuration	$N_{\text{top}} = 10^4$		$N_{\text{top}} = N_{\text{max}}$	
	$m, (10^{-3})$	R^2	$m, (10^{-3})$	R^2
Classifier	10.42	0.560	-0.74	0.848
Generator	2.37	0.363	-0.15	0.172
MPM	-0.70	0.212	-0.59	0.992
Classifier + Generator	3.49	0.454	-0.52	0.819
Classifier + MPM	-0.26	0.363	-0.51	0.666
Generator + MPM	4.64	0.738	-0.51	0.859
Classifier + Generator + MPM	-0.54	0.321	-0.46	0.954
Average (over pre-train modes)	2.78	0.430	-0.50	0.759

TABLE XV. Small model, top tagging mean test loss, and mean background rejection $1/\epsilon_{\text{bg}}$ at $\epsilon_s = 0.3$ for all pre-training modes across fine-tuning sample sizes $N_{\text{ft}} \in \{10^3, 10^4, 10^5, 10^6\}$. Bold marks the best value per column.

Configuration	$N_{\text{ft}} = 10^3$		$N_{\text{ft}} = 10^4$		$N_{\text{ft}} = 10^5$		$N_{\text{ft}} = 10^6$	
	Loss	$1/\epsilon_{\text{bg}}$	Loss	$1/\epsilon_{\text{bg}}$	Loss	$1/\epsilon_{\text{bg}}$	Loss	$1/\epsilon_{\text{bg}}$
Classifier	0.6344	291	0.5430	850	0.5402	2210	0.5400	2820
Generator	0.5935	30	0.5639	132	0.5456	739	0.5430	1200
MPM	0.5513	15	0.5436	484	0.5428	1711	0.5404	2217
From Scratch	0.6820	25	0.5525	369	0.5511	682	0.5400	2091
Classifier + Generator	0.6016	248	0.5436	937	0.5409	1766	0.5389	2781
Classifier + MPM	0.5443	556	0.5418	1375	0.5413	1607	0.5395	2820
Generator + MPM	0.5583	10	0.5477	494	0.5439	894	0.5410	1679
Classifier + Generator + MPM	0.5446	419	0.5429	1061	0.5416	1856	0.5398	2505

TABLE XVI. Medium model, top tagging mean test loss and mean background rejection $1/\epsilon_{\text{bg}}$ at $\epsilon_s = 0.3$ for all pre-training modes across fine-tuning sample sizes $N_{\text{ft}} \in \{10^3, 10^4, 10^5, 10^6\}$. Bold marks the best value per column.

Configuration	$N_{\text{ft}} = 10^3$		$N_{\text{ft}} = 10^4$		$N_{\text{ft}} = 10^5$		$N_{\text{ft}} = 10^6$	
	Loss	$1/\epsilon_{\text{bg}}$	Loss	$1/\epsilon_{\text{bg}}$	Loss	$1/\epsilon_{\text{bg}}$	Loss	$1/\epsilon_{\text{bg}}$
Classifier	0.5418	662	0.5406	1121	0.5411	2038	0.5394	3074
Generator	0.5665	7	0.5543	323	0.5458	731	0.5412	1520
MPM	0.5507	12	0.5434	24	0.5405	1659	0.5406	2399
From Scratch	0.5757	74	0.5604	219	0.5469	807	0.5415	1589
Classifier + Generator	0.5417	231	0.5406	857	0.5407	2022	0.5389	2874
Classifier + MPM	0.5430	438	0.5399	1175	0.5405	2082	0.5390	2832
Generator + MPM	0.5530	10	0.5447	260	0.5432	1253	0.5407	2063
Classifier + Generator + MPM	0.5466	165	0.5399	1007	0.5406	1957	0.5392	2792

TABLE XVII. Micro model, top tagging mean test loss and mean background rejection $1/\epsilon_{bg}$ at $\epsilon_s = 0.3$, for every pre-training objective and pre-training dataset size N_{pt} , across fine-tuning sample sizes $N_{ft} \in \{10^3, 10^4, 10^5, 10^6\}$. Configuration blocks are separated by double rules; $N_{pt} = 10^8$ denotes the full (100M) pre-training set, and From Scratch ($N_{pt} = -$) uses no pre-training. Bold marks the best value in each column across all rows.

Configuration	N_{pt}	$N_{ft} = 10^3$		$N_{ft} = 10^4$		$N_{ft} = 10^5$		$N_{ft} = 10^6$	
		Loss	$1/\epsilon_{bg}$	Loss	$1/\epsilon_{bg}$	Loss	$1/\epsilon_{bg}$	Loss	$1/\epsilon_{bg}$
Classifier	10^5	0.5554	357	0.5453	602	0.5439	1038	0.5420	1457
	10^6	0.5535	324	0.5432	653	0.5422	1135	0.5406	1894
	10^7	0.5861	131	0.5436	796	0.5420	1742	0.5398	2396
	10^8	0.6043	136	0.5800	331	0.5463	1221	0.5398	2731
Generator	10^5	0.5647	50	0.5605	188	0.5446	747	0.5424	1229
	10^6	0.5684	26	0.5585	226	0.5467	633	0.5431	1170
	10^7	0.5694	11	0.5576	210	0.5463	646	0.5428	1152
	10^8	0.6058	20	0.5687	98	0.5468	762	0.5420	1434
MPM	10^5	0.6787	19	0.5497	287	0.5455	814	0.5425	1335
	10^6	0.5754	45	0.5511	406	0.5447	766	0.5419	1350
	10^7	0.5679	91	0.5464	487	0.5448	886	0.5412	1644
	10^8	0.5562	218	0.5489	539	0.5442	899	0.5408	1793
From Scratch	–	0.6416	33	0.5699	75	0.5453	896	0.5419	1528
Classifier + Generator	10^5	0.5555	27	0.5450	505	0.5431	960	0.5417	1461
	10^6	0.5481	178	0.5443	510	0.5421	1128	0.5407	1723
	10^7	0.5530	14	0.5427	750	0.5417	1342	0.5401	1897
	10^8	0.6057	102	0.5572	517	0.5417	1411	0.5402	1996
Classifier + MPM	10^5	0.5620	211	0.5462	603	0.5444	939	0.5420	1365
	10^6	0.5531	437	0.5450	450	0.5423	1077	0.5407	1761
	10^7	0.5535	311	0.5456	269	0.5424	1188	0.5402	1963
	10^8	0.5569	356	0.5451	675	0.5428	1137	0.5405	2006
Generator + MPM	10^5	0.6366	44	0.5531	297	0.5451	822	0.5425	1335
	10^6	0.6009	76	0.5568	263	0.5450	788	0.5424	1376
	10^7	0.5650	62	0.5559	280	0.5454	760	0.5420	1375
	10^8	0.5855	32	0.5689	131	0.5458	789	0.5410	1647
Classifier + Generator + MPM	10^5	0.5517	117	0.5480	542	0.5446	899	0.5414	1612
	10^6	0.5531	17	0.5451	434	0.5429	1058	0.5410	1897
	10^7	0.5549	14	0.5457	290	0.5427	1033	0.5403	1997
	10^8	0.5563	255	0.5459	719	0.5436	1161	0.5401	2028

TABLE XVIII. Medium model, classifier pre-training, downstream top tagging as a function of pre-training checkpoint along the pre-training trajectory. Each row is one pre-training checkpoint fine-tuned on the full 1.2M-Top set. Bold marks the best value in each metric column.

Pre-train step	Pre-train val loss	Finetune val loss	$1/\epsilon_{\text{bg}} (\epsilon_s=0.3)$
2500	0.7666	0.1524	1697
14707	0.4501	0.1445	2243
26914	0.4299	0.1431	2766
39121	0.4198	0.1422	3059
51328	0.4160	0.1417	2969
63535	0.4139	0.1415	3106
75742	0.4119	0.1413	2926
87949	0.4102	0.1411	3106
100156	0.4097	0.1411	3106
112363	0.4082	0.1413	2804
124570	0.4062	0.1414	2804
136777	0.4039	0.1415	2657
148984	0.4070	0.1419	2657
161191	0.4033	0.1422	2969
175898	0.4036	0.1419	2556
188105	0.4053	0.1422	2926
200312	0.4057	0.1423	2556
212519	0.4089	0.1423	2804
224726	0.4095	0.1426	3014
236933	0.4126	0.1428	2729
249140	0.4134	0.1430	2589
261347	0.4185	0.1431	2524

TABLE XIX. Medium model, classifier+generation pre-training, downstream top tagging as a function of pre-training checkpoint along the pre-training trajectory. Each row is one pre-training checkpoint fine-tuned on the full 1.2M-Top set. Bold marks the best value in each metric column.

Pre-train step	Pre-train val loss	Finetune val loss	$1/\epsilon_{\text{bg}} (\epsilon_s=0.3)$
2500	4.1142	0.1523	1383
7500	3.4389	0.1487	1787
14707	3.3342	0.1456	2060
17207	3.3287	0.1451	2348
19707	3.3203	0.1446	2348
26914	3.2925	0.1440	2375
31914	3.2649	0.1435	2524
39121	3.2578	0.1427	2524
44121	3.2540	0.1429	2729
46621	3.2477	0.1427	2462
51328	3.2561	0.1426	2729
56328	3.2391	0.1421	2556
63535	3.2368	0.1419	2657
68535	3.2386	0.1421	2804
75742	3.2343	0.1418	2884
78242	3.2320	0.1416	2729
80742	3.2235	0.1417	3257
87949	3.2286	0.1413	2729
92949	3.2244	0.1416	2766
100156	3.2258	0.1412	3205
107656	3.2220	0.1411	2884
124570	3.2202	0.1411	2692
139277	3.2213	0.1410	2884
153984	3.2106	0.1410	2884
163691	3.2150	0.1412	2729
168691	3.2133	0.1410	2589
175898	3.2116	0.1410	3014
178398	3.2074	0.1411	3014
180898	3.2106	0.1412	2884
188105	3.2096	0.1412	3014
193105	3.2072	0.1409	2729
200312	3.2088	0.1411	3106
205312	3.2100	0.1410	3059
210019	3.2100	0.1411	3155
212519	3.2119	0.1412	2969
222226	3.2125	0.1412	3257
227226	3.2039	0.1411	2766
234433	3.2106	0.1415	3014
239433	3.2081	0.1414	2766
241933	3.2074	0.1413	2729

TABLE XX. Performance of the different generative models averaged over all jet types. Each model is trained for 520 epochs on the full dataset ($N_{\text{JetNet}} = 550\text{k}$).

(a) Medium							
	Jet p_T (GeV)	Jet η $\times 10^{-3}$	$\min(p_i p_j)$ $\times 10^{-5}$	$\Delta\eta_i$ $\times 10^{-4}$	$\Delta\phi_i$ $\times 10^{-4}$	$\log p_{T,i}$ $\times 10^{-3}$	$\log E_i$ $\times 10^{-3}$
From Scratch	2.4 ± 0.8	8.0 ± 1.3	3.6 ± 1.1	9.1 ± 1.1	8.8 ± 1.5	4.8 ± 0.9	6.4 ± 1.2
Generator	2.4 ± 0.5	8.5 ± 1.2	4.9 ± 1.5	9.0 ± 1.3	8.4 ± 1.4	5.9 ± 1.4	5.4 ± 1.1
Classifier + Generator + MPM	2.3 ± 0.6	8.5 ± 1.2	4.7 ± 1.4	8.3 ± 1.1	9.0 ± 1.3	4.5 ± 0.5	5.8 ± 1.0
Classifier + Generator	2.0 ± 0.5	8.6 ± 1.6	4.2 ± 1.5	8.6 ± 1.2	9.3 ± 1.2	5.7 ± 1.6	7.5 ± 1.9
Generator + MPM	2.3 ± 0.8	7.2 ± 1.0	4.8 ± 1.5	8.3 ± 1.0	8.5 ± 1.5	5.0 ± 1.1	6.4 ± 1.5
Classifier	2.6 ± 0.5	9.4 ± 1.5	4.4 ± 1.3	11.8 ± 1.4	11.7 ± 1.3	4.3 ± 0.7	6.8 ± 2.3
MPM	2.1 ± 0.5	9.0 ± 1.4	3.7 ± 1.5	10.7 ± 1.4	10.4 ± 1.4	4.9 ± 0.7	6.8 ± 1.5
Classifier + MPM	2.5 ± 0.3	9.8 ± 1.1	3.8 ± 1.2	12.1 ± 1.0	12.5 ± 1.2	5.0 ± 0.8	5.3 ± 0.8

(b) Small							
	Jet p_T (GeV)	Jet η $\times 10^{-3}$	$\min(p_i p_j)$ $\times 10^{-5}$	$\Delta\eta_i$ $\times 10^{-4}$	$\Delta\phi_i$ $\times 10^{-4}$	$\log p_{T,i}$ $\times 10^{-3}$	$\log E_i$ $\times 10^{-3}$
From Scratch	2.3 ± 0.5	8.0 ± 1.1	4.4 ± 1.3	7.5 ± 1.2	7.1 ± 0.8	4.6 ± 0.7	6.1 ± 1.3
Generator	2.2 ± 0.7	8.7 ± 2.1	4.8 ± 1.3	7.8 ± 0.9	7.2 ± 1.0	5.8 ± 1.1	5.8 ± 1.0
Classifier + Generator + MPM	2.3 ± 0.5	7.4 ± 1.1	5.0 ± 1.3	6.9 ± 0.9	7.0 ± 1.4	5.5 ± 1.1	5.7 ± 1.0
Classifier + Generator	2.2 ± 0.6	7.6 ± 1.8	5.3 ± 1.9	7.2 ± 0.9	7.6 ± 1.0	4.6 ± 1.0	6.5 ± 1.3
Generator + MPM	2.1 ± 0.5	8.3 ± 1.6	5.1 ± 1.7	7.2 ± 1.1	6.4 ± 1.0	6.0 ± 1.2	7.5 ± 1.9
Classifier	2.3 ± 0.7	7.5 ± 1.0	4.5 ± 1.4	7.7 ± 0.9	7.4 ± 1.0	6.5 ± 1.5	7.0 ± 2.3
MPM	2.2 ± 0.5	8.6 ± 1.8	3.6 ± 1.1	7.1 ± 1.0	7.8 ± 1.2	5.9 ± 1.4	6.4 ± 1.6
Classifier + MPM	2.2 ± 0.7	7.3 ± 1.1	4.0 ± 1.2	8.8 ± 1.3	8.9 ± 1.4	5.4 ± 1.2	5.8 ± 1.4

(c) Micro							
	Jet p_T (GeV)	Jet η $\times 10^{-3}$	$\min(p_i p_j)$ $\times 10^{-5}$	$\Delta\eta_i$ $\times 10^{-4}$	$\Delta\phi_i$ $\times 10^{-4}$	$\log p_{T,i}$ $\times 10^{-3}$	$\log E_i$ $\times 10^{-3}$
From Scratch	2.0 ± 0.3	10.3 ± 1.2	3.3 ± 0.8	14.7 ± 1.7	20.2 ± 2.1	5.5 ± 1.2	7.4 ± 1.3
Generator	3.2 ± 0.4	9.7 ± 1.7	3.3 ± 1.3	36.8 ± 3.5	36.9 ± 2.7	6.9 ± 1.4	7.2 ± 1.3
Classifier + Generator + MPM	3.1 ± 0.5	12.8 ± 2.1	6.1 ± 1.8	43.0 ± 6.6	53.7 ± 16.1	6.2 ± 0.9	6.9 ± 2.1
Classifier + Generator	9.6 ± 0.9	13.4 ± 2.1	5.8 ± 1.7	22.4 ± 2.7	21.5 ± 2.9	9.3 ± 2.2	8.4 ± 2.3
Generator + MPM	3.1 ± 0.5	12.0 ± 2.0	3.3 ± 0.9	33.8 ± 1.9	34.9 ± 2.3	6.1 ± 1.2	7.1 ± 1.6
Classifier	3.0 ± 0.5	11.6 ± 1.3	5.9 ± 1.4	31.7 ± 4.4	33.5 ± 5.1	5.7 ± 0.7	6.8 ± 1.4
MPM	2.1 ± 0.3	9.4 ± 1.5	3.8 ± 1.0	22.2 ± 2.8	23.8 ± 2.4	5.3 ± 0.9	6.5 ± 1.6
Classifier + MPM	2.5 ± 0.7	8.5 ± 1.4	3.6 ± 0.9	17.5 ± 2.4	20.3 ± 3.7	5.1 ± 0.6	5.8 ± 1.4

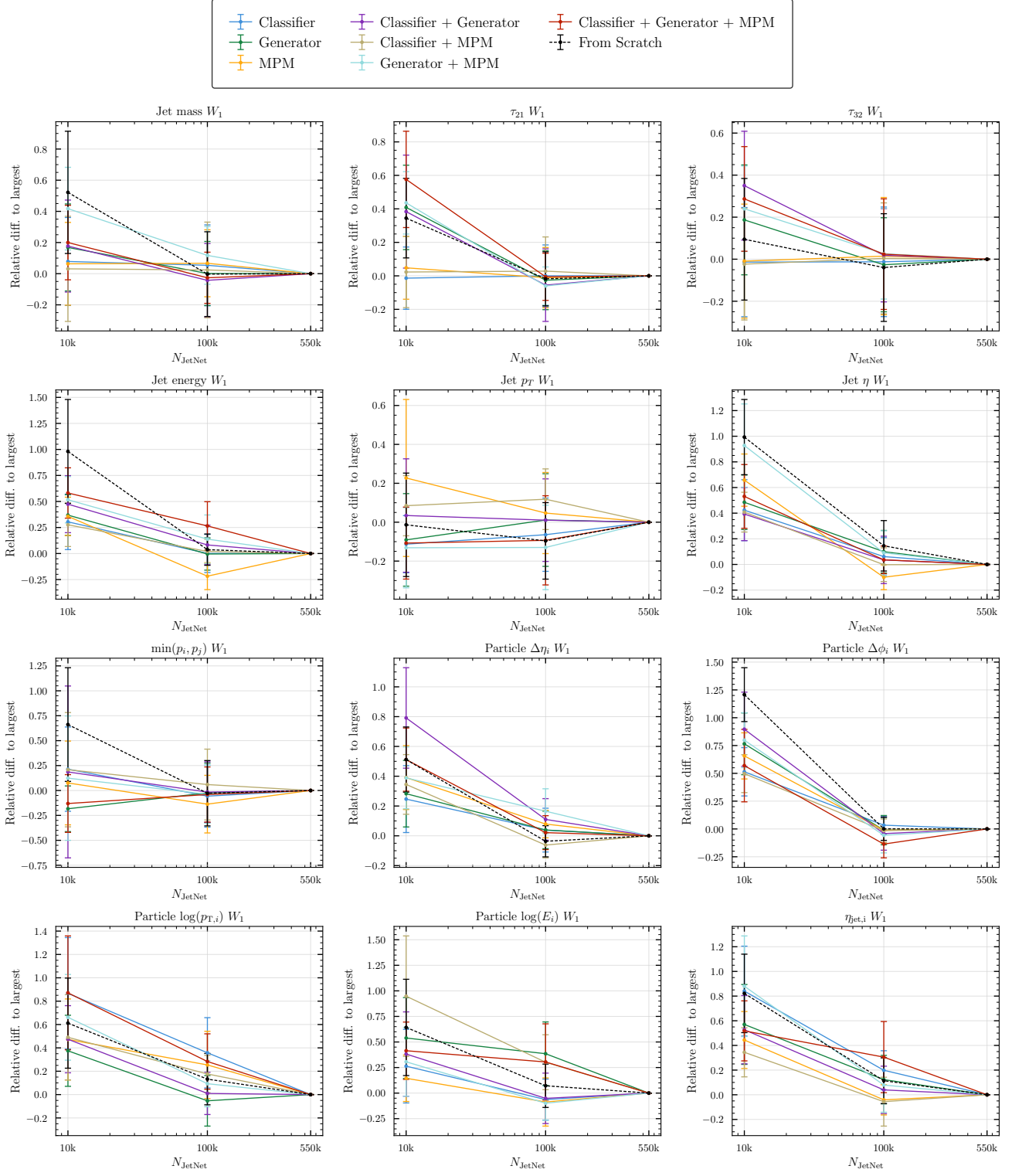


FIG. 18. Relative performance difference of the generative model as a function of the training dataset size compared to the corresponding model obtained with the full dataset size ($N_{\text{JetNet}} = 550\text{k}$) for the medium model. The relative performance difference is given by $(W_1(N_{\text{JetNet}}) - W_1(550\text{k})) / W_1(550\text{k})$ for the corresponding metric.

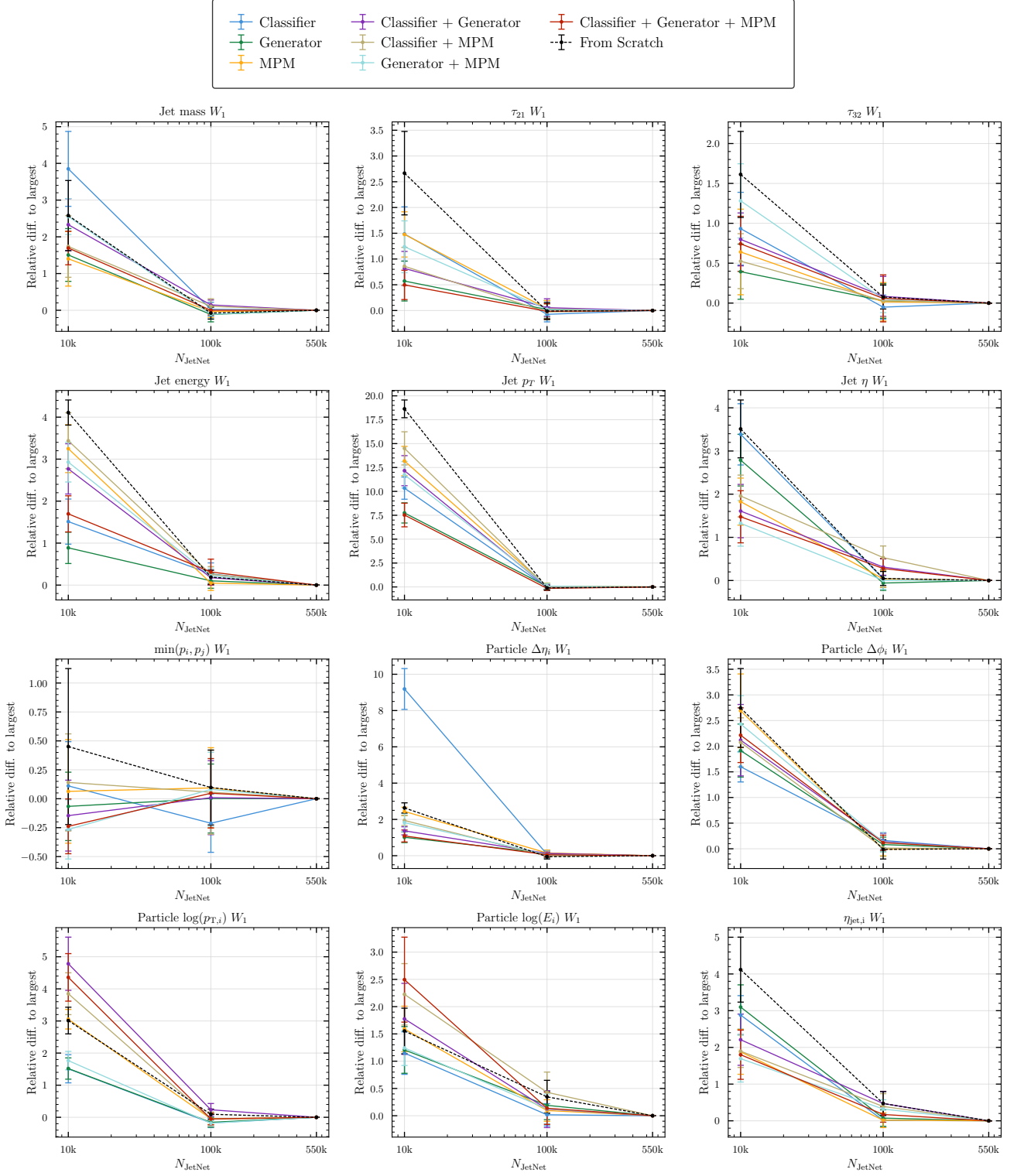


FIG. 19. Relative performance difference of the generative model as a function of the training dataset size compared to the corresponding model obtained with the full dataset size ($N_{\text{JetNet}} = 550\text{k}$) for the small model. The relative performance difference is given by $(W_1(N_{\text{JetNet}}) - W_1(550\text{k})) / W_1(550\text{k})$ for the corresponding metric.

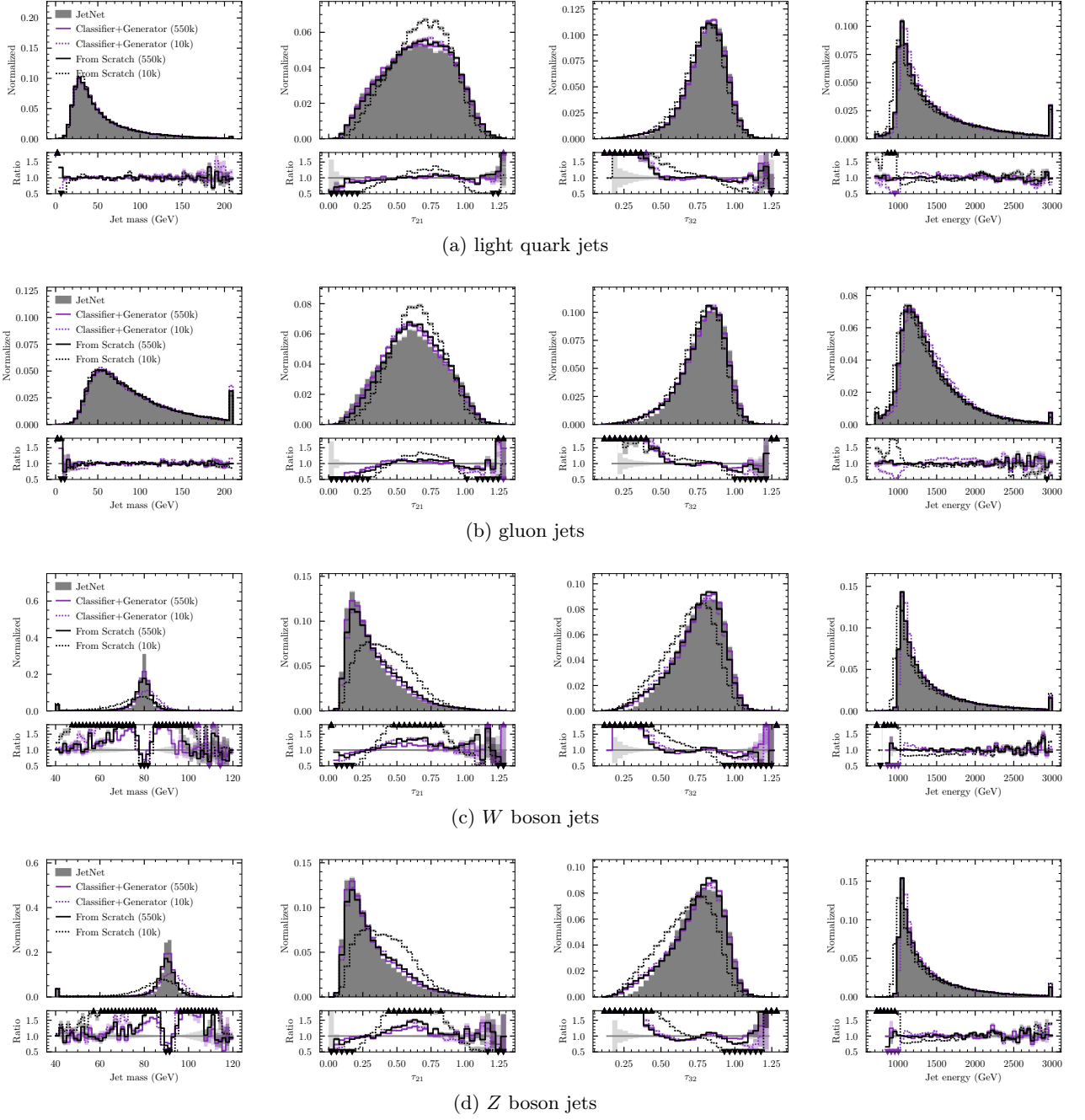


FIG. 20. Jet-level distributions of jets initiated by (a) light quarks, (b) gluons, (c) W bosons and (d) Z bosons. Generated jets are obtained from the small model with the Classifier+Generator pre-training method compared with the from scratch baseline. Two downstream dataset sizes are shown in dotted ($N_{\text{JetNet}} = 10\text{k}$) and solid ($N_{\text{JetNet}} = 550\text{k}$) lines. The gray histogram shows jets from the JetNet dataset.

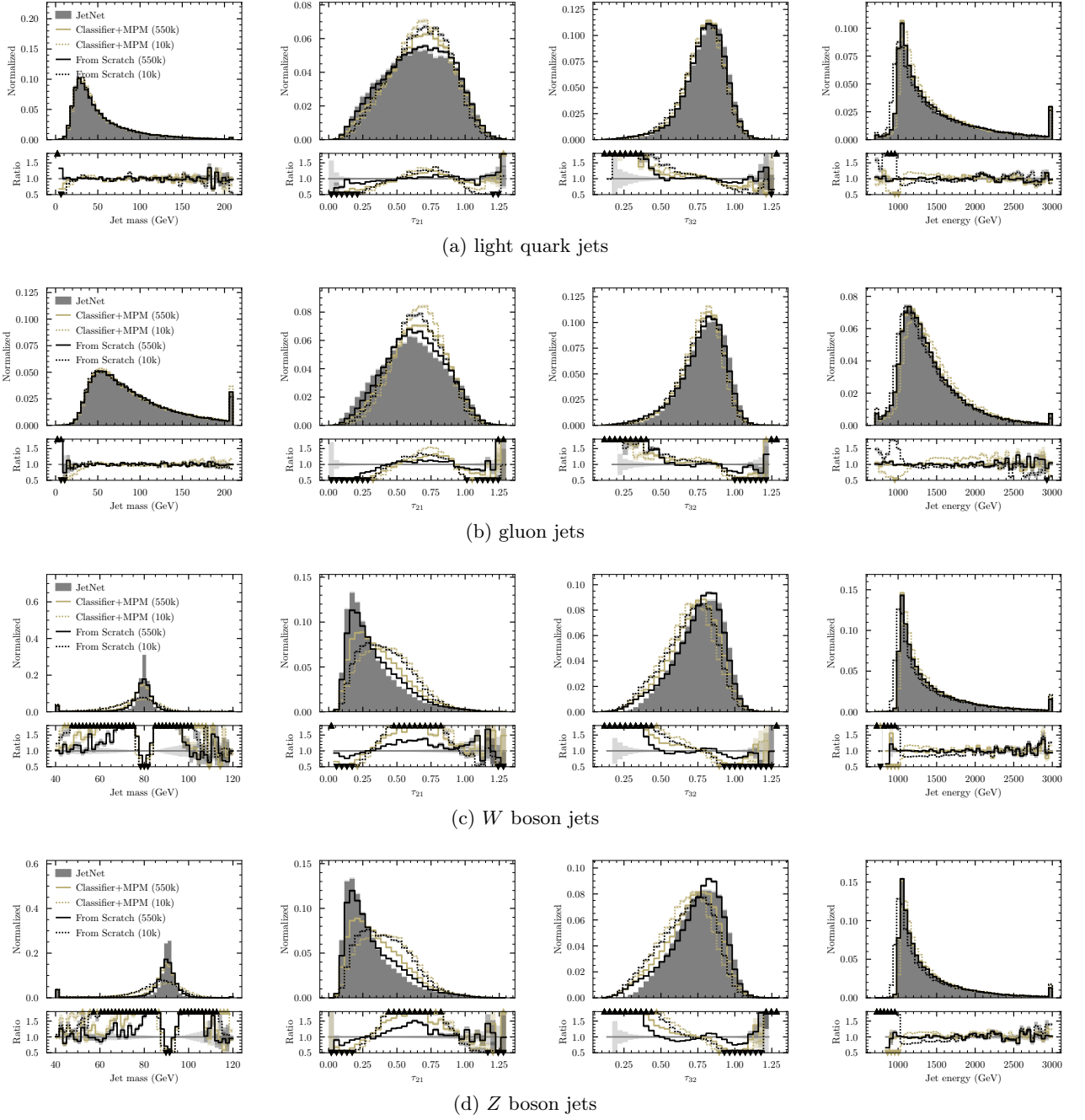


FIG. 21. Jet-level distributions of jets initiated by (a) light quarks, (b) gluons, (c) W bosons and (d) Z bosons. Generated jets are obtained from the small model with the Classifier+MPM pre-training method compared with the from scratch baseline. Two downstream dataset sizes are shown in dotted ($N_{\text{JetNet}} = 10\text{k}$) and solid ($N_{\text{JetNet}} = 550\text{k}$) lines. The gray histogram shows jets from the JetNet dataset.

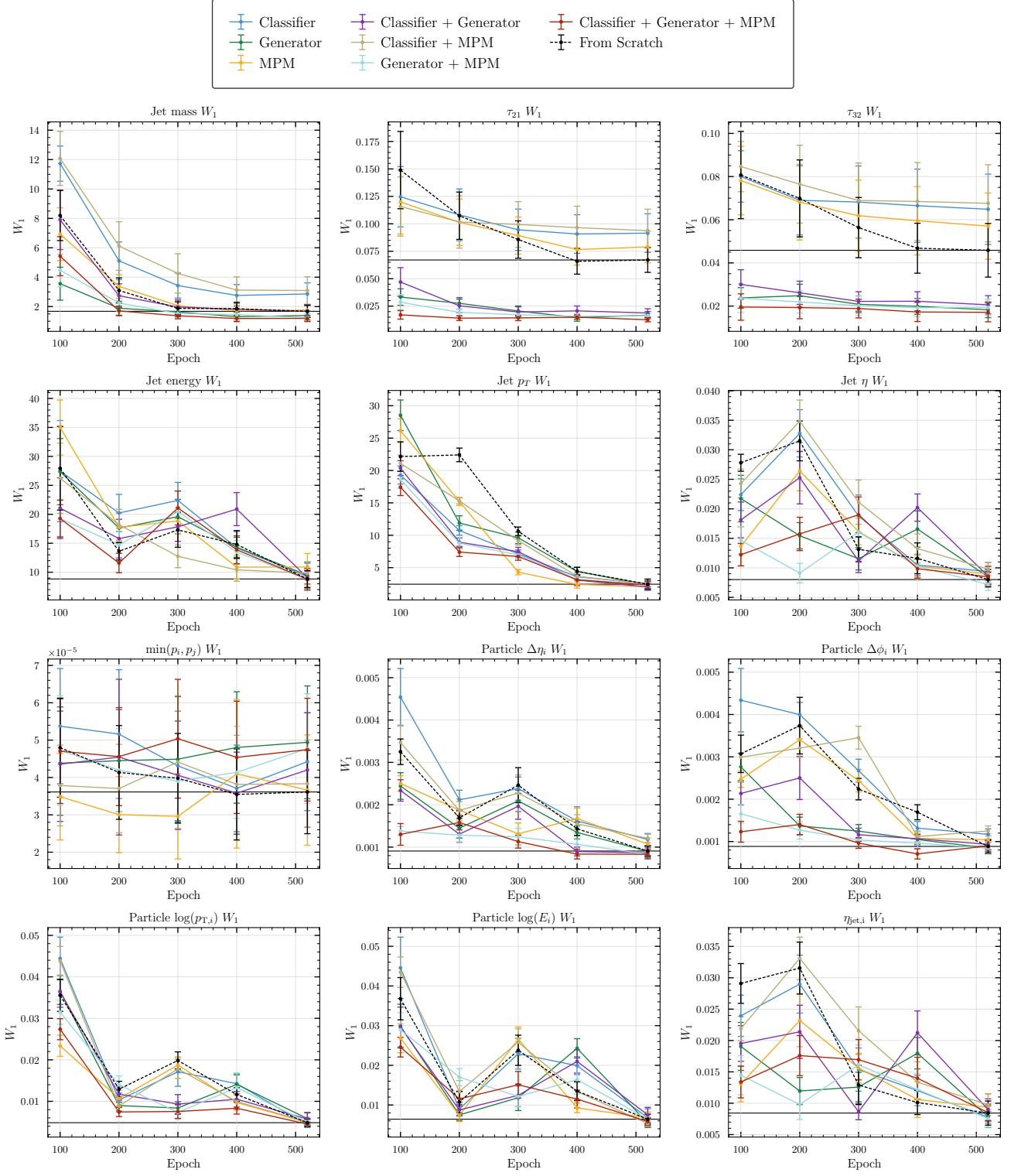


FIG. 22. Performance of the generative models as a function of the training epoch using a training dataset size $N_{\text{JetNet}} = 550\text{k}$ for the medium model size. The black horizontal line marks the value corresponding to the final epoch of the from scratch baseline.

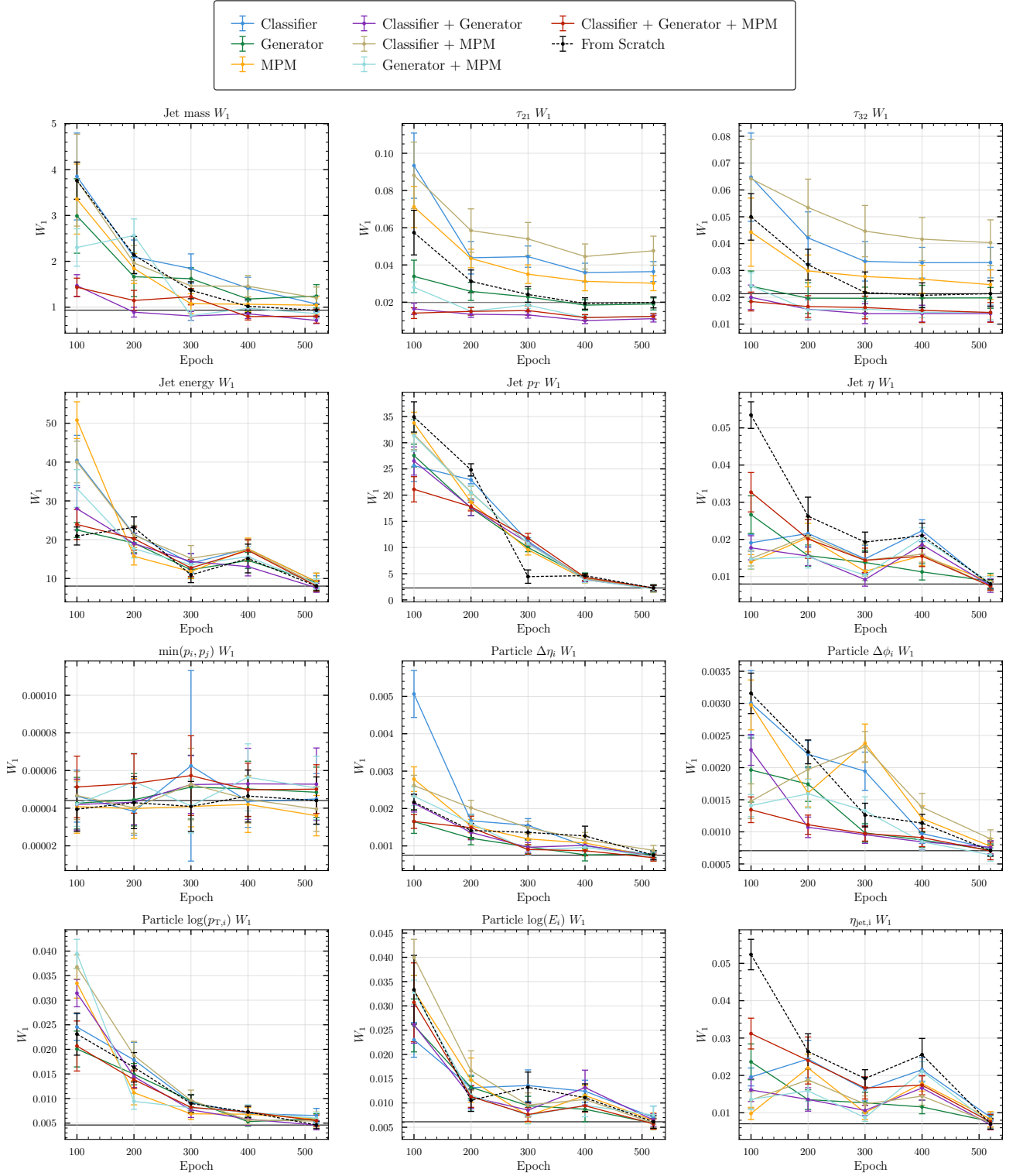


FIG. 23. Performance of the generative models as a function of the training epoch using a training dataset size $N_{\text{JetNet}} = 550\text{k}$ for the small model size. The black horizontal line marks the value corresponding to the final epoch of the from scratch baseline.

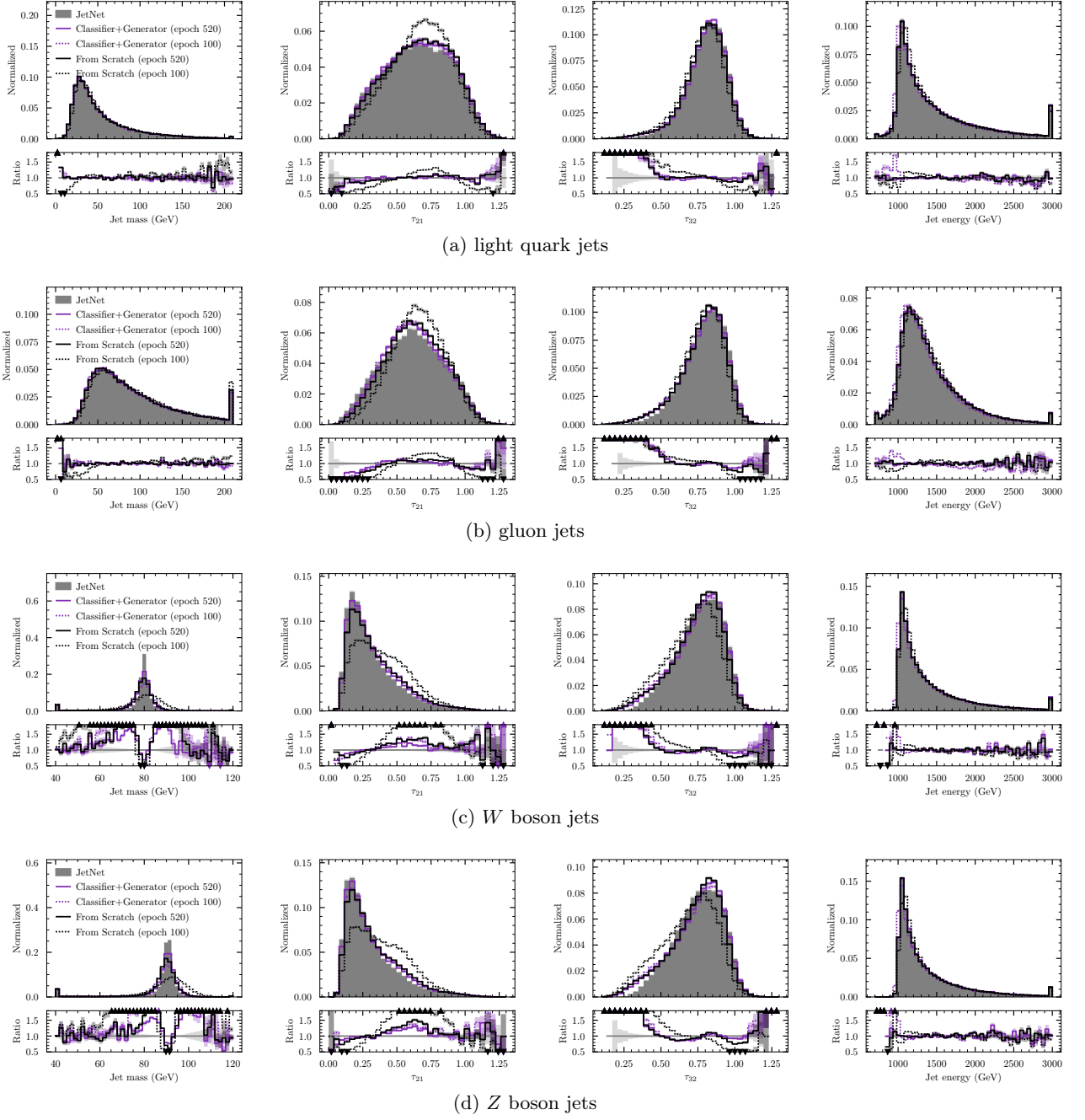


FIG. 24. Jet-level distributions of jets initiated by (a) light quarks, (b) gluons, (c) W bosons and (d) Z bosons. Generated jets are obtained from the small model with the Classifier+Generator pre-training method compared with the from scratch baseline. Two evaluated epochs are shown in dotted (epoch 100) and solid (epoch 520) lines. The gray histogram shows jets from the JetNet dataset.

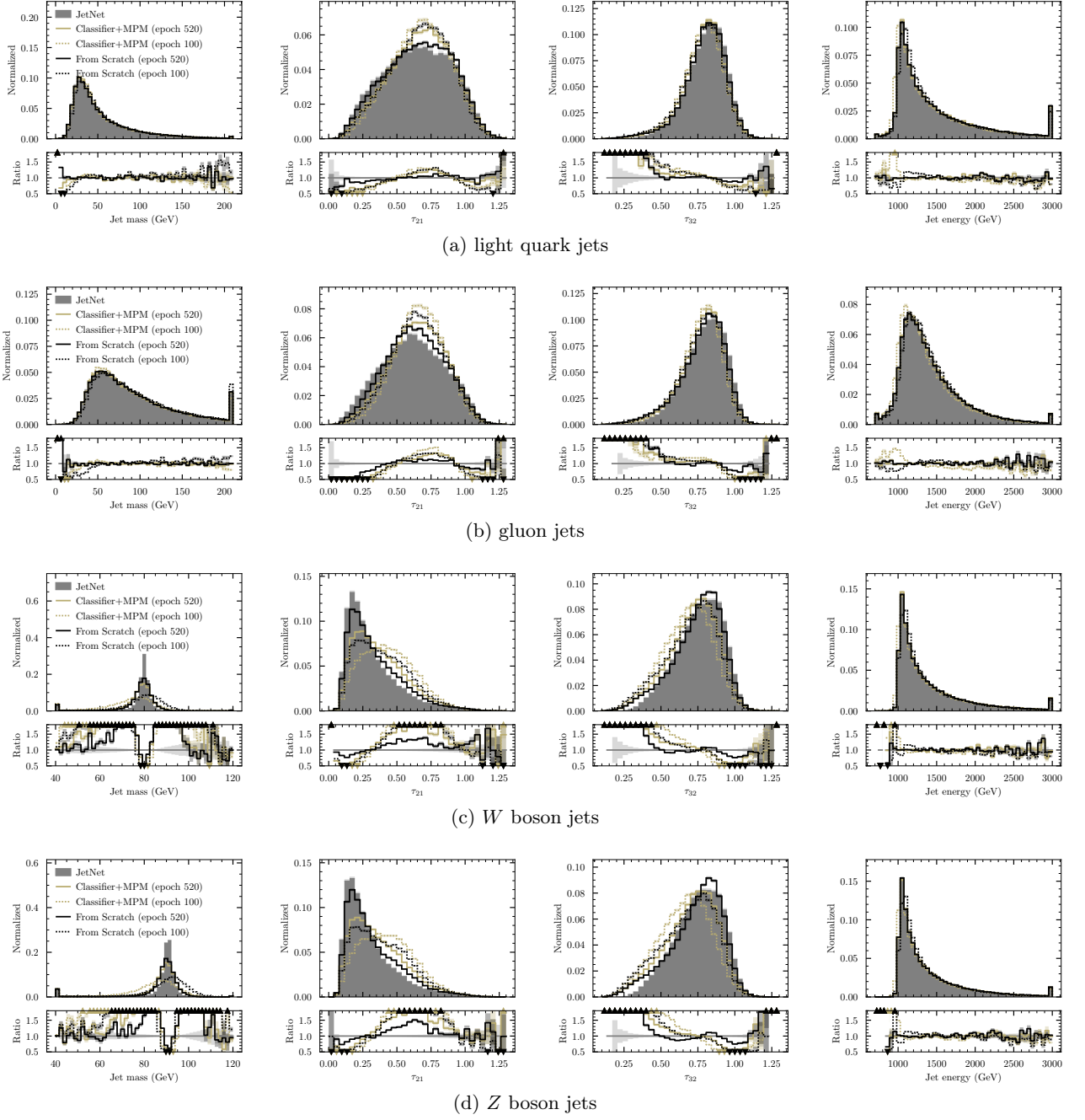


FIG. 25. Jet-level distributions of jets initiated by (a) light quarks, (b) gluons, (c) W bosons and (d) Z bosons. Generated jets are obtained from the small model with the Classifier+MPM pre-training method compared with the from scratch baseline. Two evaluated epochs are shown in dotted (epoch 100) and solid (epoch 520) lines. The gray histogram shows jets from the JetNet dataset.