Fast point cloud generation with diffusion models in high energy physics

Vinicius Mikuni,[1,*] Benjamin Nachman,[2,3,†] and Mariel Pettee[2,‡]

[1]*National Energy Research Scientific Computing Center, Berkeley Lab, Berkeley, California 94720, USA*
[2]*Physics Division, Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA*
[3]*Berkeley Institute for Data Science, University of California, Berkeley, California 94720, USA*

(Received 28 April 2023; accepted 8 August 2023; published 28 August 2023)

Many particle physics datasets like those generated at colliders are described by continuous coordinates (in contrast to grid points like in an image), respect a number of symmetries (like permutation invariance), and have a stochastic dimensionality. For this reason, standard deep generative models that produce images or at least a fixed set of features are limiting. We introduce a new neural network simulation based on a diffusion model that addresses these limitations named fast point cloud diffusion. We show that our approach can reproduce the complex properties of hadronic jets from proton-proton collisions with competitive precision to other recently proposed models. Additionally, we use a procedure called progressive distillation to accelerate the generation time of our method, which is typically a significant challenge for diffusion models despite their state-of-the-art precision.

DOI: 10.1103/PhysRevD.108.036025

I. INTRODUCTION

Simulations are a critical component of nearly all inference tasks in particle physics. These simulations connect theory to experiment and must span a wide range of energy scales and encode the complex structure of high energy physics data. Physics-based simulations are excellent, but they are only an approximation to nature. Additionally, some components of these simulations are computationally expensive and are a bottleneck for the high statistics datasets that are being collected now and in the near future. Classical fast approximations exist for some steps and in some cases, such as detector simulations for a particular experiment, but they are often not expressive enough to achieve high fidelity compared to a full simulation routine.

Deep neural network-based simulations (called deep generative models) are a promising alternative to classical fast simulations. Since the first deep generative model applied to high energy physics [1], there have been a large number of proposals to use these tools for fast simulation and many other applications [2–4]. In this paper, we revisit the original problem of emulating parton shower Monte Carlo simulations. These simulations describe the formation of jets of hadrons that emerge from the high energy quarks and gluons. Jets are ubiquitous at particle colliders and are the most complex objects reconstructed from hadronic final states. Together, these qualities make jets a standard benchmark for developing machine learning-based generative models.

Many deep generative models have been deployed to the problem of emulating jet formation. The first approaches used images by spatially discretizing the radiation pattern within jets [5,6]. Generative adversarial networks (GANs) [1,7] and autoencoders [8] were able to reproduce many aspects of the parton shower, but they were fundamentally limited because of their pixelization. While other applications of deep generative models naturally process image data (e.g., calorimeter simulations [9–33]), jets are naturally represented as variable-sized point clouds and information is lost when they are projected onto fixed size grids with reduced spatial position information compared to the original detector granularity.

Point cloud generative models (PCGM) offer the solution to the inherent challenges with pixelation. The first PCGM applied to jet formation was Ref. [34], which used a recurrent model to describe the probability density of a given jet. Recently, there has been a surge of interest in more general PCGMs that do not need to make any assumptions or approximations about the underlying generative process. This latest wave of methods began with a graph neural network-based GAN [35] and now includes a deep-sets-based GAN [36] and a normalizing flow [37,38]. These models mark a significant step forward in the application of generative models to particle physics, but

[*]vmikuni@lbl.gov
[†]bpnachman@lbl.gov
[‡]mpettee@lbl.gov

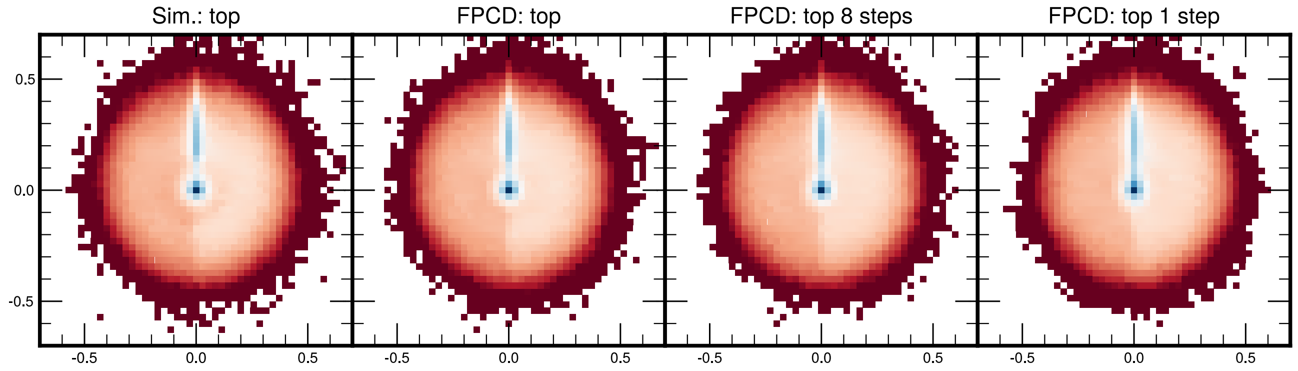

FIG. 1. Average top quark initiated jet in the full simulation, after generation with the diffusion model, and after distillation resulting in eight or a single time step used during sampling.

there is still significant room for improvement in both precision and robustness. For example, GANs solve a minimax problem and are thus difficult to train. Normalizing flows are more stable to train, but may have difficulties when generating low level inputs (such as particle kinematic information), or a variable-length representation with complex topology due to the invertible nature of their neural networks.

In the machine learning literature, the most precise generative neural networks are diffusion models (see, e.g., Ref. [39]). These approaches circumvent the challenges with other models by performing a convex optimization problem, but without the need for invertible transformations. This can be achieved by learning the score of the probability density $\nabla \log p$ instead of the probability density directly. The first diffusion model applied to particle physics was in the context of image-based calorimeter simulation [33], significantly extending the dimensionality of previous results. Our goal is to adapt diffusion models to the variable-length point cloud setting for parton showers and other phenomena in high energy physics while also reducing the generation time to be competitive with other fast generation methods. To this end, we introduce our algorithm for fast point cloud generation (FPCD), used to simulate point cloud data with varying length much faster than a standard diffusion implementation. Examples of generated point clouds using our proposed algorithm are shown in Fig. 1, where we compare the average energy deposition for top quark initiated jets generated by the full simulation or by the generative model. We accelerate the sampling time of the surrogate model using a method called progressive distillation [40], resulting in a generative model with high physics fidelity and fast sampling times.

While this paper was being finalized, the authors of Ref. [41] also proposed a diffusion-based PCGM for jet formation. The proposal in our paper differs from Ref. [41] in a few ways. First, our model does not condition on the jet mass, but rather it utilizes a separate diffusion model to determine the jet kinematics. Next, it is much faster (via progressive distillation) and is conditioned on the particle type, thereby avoiding the training of multiple diffusion models for each type of jet. Finally, we also provide results for more particle types (including gluons and W and Z bosons in addition to light and top quarks) in two different datasets with varying number of particles to demonstrate that our model is capable of generating outputs of varying sizes.

This paper is organized as follows: Sec. II introduces score-based diffusion models and describes how they can be accelerated with progressive distillation. We then detail our implementation of the diffusion-based generative model for parton showers in Sec. III. Numerical results are presented in Sec. IV and the paper ends with conclusions and outlook in Sec. V.

II. SCORE-BASED GENERATIVE MODELS AND PROGRESSIVE DISTILLATION

The goal of a generative model is to be able to generate new observations from a noise distribution. Diffusion models became popular in recent years for their capacity to generate realistic data, often surpassing standard state-of-the-art generative models. In score-based methods [42], a diffusion process is designed to slowly perturb the data through the addition of noise, while a neural network learns a time-dependent score function $\nabla_x \log p_{\mathrm{data}}$ for some high-dimensional distribution $\mathbf{x} \in \mathbb{R}^D$ described by the probability density p_{data}. The score function is then used in a reverse-diffusion process: starting from a noisy distribution and proceeding to denoise the observation. The diffusion model is described by latent variables $\mathbf{z} = \{\mathbf{z}_t | t \in [0,1]\}$ with a time-dependent noise schedule α_t, σ_t, such that the log signal-to-noise ratio $\log[\alpha_t^2/\sigma_t^2]$, decreases monotonically with time. During training, the network learns to denoise $\mathbf{z}_t \sim q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I})$ towards the unperturbed data $\mathbf{x} \sim p_{\mathrm{data}}$, effectively learning an estimate $\hat{\mathbf{x}}_\theta \approx \mathbf{x}$ by updating the trainable parameters θ during training. Following Ref. [40], we instead train a network to estimate a "velocity" parameter $\mathbf{v} \equiv \alpha_t \epsilon - \sigma_t \mathbf{x}$, with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, which is observed to yield accurate results while also simplifying the distillation method employed later. The loss function to be minimized during optimization is then defined as

$$\mathcal{L}_\theta = \mathbb{E}_{\epsilon,t} \|\mathbf{v}_t - \hat{\mathbf{v}}_{t,\theta}\|^2, \tag{1}$$

where t is sampled uniformly over the considered interval. In this formulation, we can identify the estimate of the score function as

$$\nabla_z \log \hat{p}_\theta(\mathbf{z}_t) = \mathbf{z}_t - \frac{\alpha_t}{\sigma_t}\hat{\mathbf{v}}_\theta(\mathbf{z}_t). \tag{2}$$

In our implementation, we consider the variance-preserving setting of diffusion processes, where $\sigma_t^2 = 1 - \alpha_t^2$. For the time dependence, we use a cosine schedule such that $\alpha_t = \cos(0.5\pi t)$.

The generation of new samples is then carried out using the Denoising Diffusion Implicit Models (DDIM) sampler proposed in Ref. [43] that uses an integration rule to solve the deterministic ordinary differential equation:

$$\mathrm{d}\mathbf{z}_t = \left[f(\mathbf{z}_t, t) - \frac{1}{2}g^2(t)\nabla_z \log \hat{p}_\theta(\mathbf{z}_t)\right]\mathrm{d}t, \tag{3}$$

with drift coefficient $f(\mathbf{z}_t, t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\mathbf{z}_t$ and diffusion coefficient $g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}$. In the DDIM solver, the update rule is then specified by

$$\mathbf{z}_s = \alpha_s \hat{\mathbf{x}}_\theta(\mathbf{z}_t) + \sigma_s \frac{\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\theta(\mathbf{z}_t)}{\sigma_t}. \tag{4}$$

In practice, solving Eq. (3) can be slow since the error introduced by the numerical integration is sensitive to the number of time steps chosen, often requiring hundreds to thousands of time steps and hence function evaluations of the trained model.

To accelerate diffusion models, Ref. [40] introduced a technique called progressive distillation. Starting from a trained diffusion model, the goal of progressive distillation is to learn iteratively to halve the number of time steps required during generation of new samples. In this setting, the trained diffusion model ("teacher") is used to initialize a "student" model. During training, the goal is to have the student model learn how to denoise data $\mathbf{z}_t$ towards a target $\tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}$ does not represent the clean data ($\mathbf{x}$) anymore, but instead is one that makes a single student DDIM step to match two teacher DDIM steps. This process is then repeated multiple times, with the student at the end of each iteration becoming the new teacher. In this work, we train a diffusion model with initial number of steps N fixed to 512. From there, we distill the model multiple times, reporting the results obtained with $N = 512$, $N = 8$, and $N = 1$.

III. POINT CLOUD DIFFUSION FOR COLLIDER DATA

particle jets conditioned on the initial particle type. We use the datasets introduced in Ref. [35] consisting of jets initiated by light quarks, gluons, top quarks, W and Z bosons. The jets are generated with transverse momenta $\mathrm{p_T}$ around 1 TeV and are clustered using the anti-k_t algorithm [44] with a radius parameter of 0.8. Each jet has a maximum number of particles stored fixed to 30 [45] or 150 [46]. For each jet, the four-momentum information $(\mathrm{p_{Tjet}}, \eta_\mathrm{jet}, \phi_\mathrm{jet}, m_\mathrm{jet})$ is provided, as well as the particle multiplicity. For each particle clustered inside a jet, the relative set of kinematic quantities are provided

$$\begin{aligned}
\mathrm{p_{Trel}} &= \mathrm{p_{Tpart}}/\mathrm{p_{Tjet}},\\
\eta_\mathrm{rel} &= \eta_\mathrm{jet} - \eta_\mathrm{part},\\
\phi_\mathrm{rel} &= \phi_\mathrm{jet} - \phi_\mathrm{part}.
\end{aligned} \tag{5}$$

Our goal is to develop a diffusion model that is conditioned on the particle's type and is able to generate both jet- and particle-level kinematic information. To accomplish this task, we train two diffusion models simultaneously. The first model learns the jet kinematic information, including particle multiplicity, while the second is conditioned on the jet kinematic distributions to generate particle information. Effectively, the loss function that is minimized during training is

$$\mathcal{L}_\mathrm{jet,particle} = \mathcal{L}_\mathrm{jet} + \mathcal{L}_\mathrm{particle}, \tag{6}$$

with different models trained to generate jet information and particle information. During the generation step, we first sample the jet kinematic information together with the particle multiplicity, conditioned on the type of the jet we aim to generate. This information is then used as an input to generate the particle information for each jet. The particle multiplicity generated determines the total number of particles generated in each jet. Although it is feasible to achieve a perfect match between the output particle multiplicity and the sampled particle multiplicity, we prefer to employ a masking plus zero-padding approach. This involves always sampling a set number of particles (either 30 or 150 depending on the dataset), but in the generation process, we mask the input noise and only consider the desired particle multiplicity.

Prior to training, the inputs to the diffusion model undergo a normalization process where all input features are standardized by adjusting their mean and standard deviation to 0 and 1, respectively.

The generative model designed to produce jet kinematic information is based on a fully connected architecture incorporating multiple skip connections. Specifically, the model employs five ResNet [47] blocks, where each residual layer is connected to the output of a two-layer network through a skip connection. The activation function used is LeakyRelu [48] with a slope of $\alpha = 0.01$, and all layer sizes are set to 512.

The particle diffusion model employs a DeepSets [49] architecture with transformer layers [50] to increase the model's expressivity. The input sets are first mapped into a larger latent space using a fully connected layer with a size of 64, applied independently to each particle in the set. The model then employs eight transformer encoding blocks

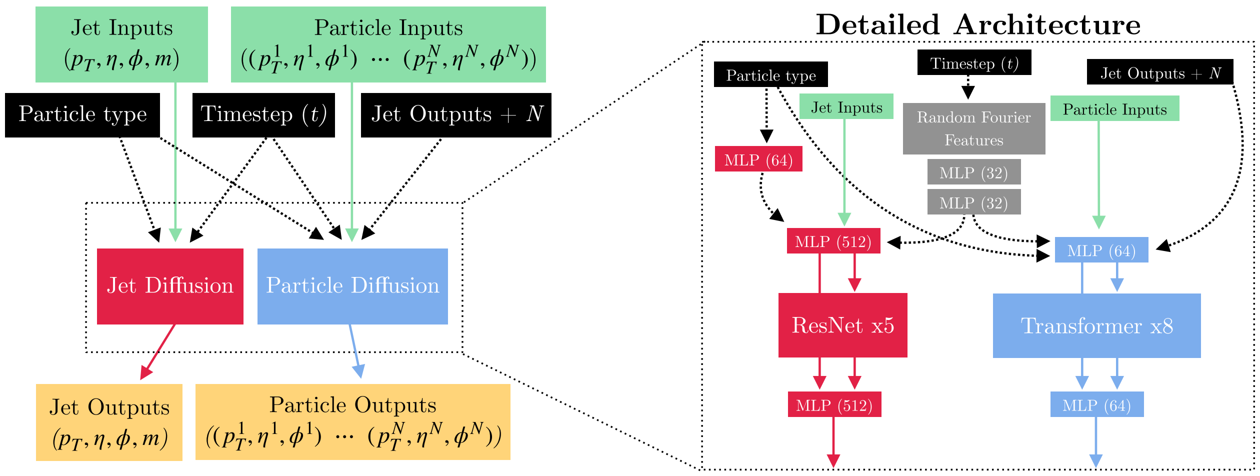
Jet Inputs (p_T, η, ϕ, m)
Particle Inputs $((p_T^1, \eta^1, \phi^1) \cdots (p_T^N, \eta^N, \phi^N))$
Particle type
Timestep (t)
Jet Outputs + N
Jet Diffusion
Particle Diffusion
Jet Outputs (p_T, η, ϕ, m)
Particle Outputs $((p_T^1, \eta^1, \phi^1) \cdots (p_T^N, \eta^N, \phi^N))$
Detailed Architecture
Random Fourier Features
MLP (64)
MLP (32)
MLP (32)
MLP (512)
MLP (64)
ResNet x5
Transformer x8
MLP (512)
MLP (64)

FIG. 2. Description of the network architectures used to train the jet and particle diffusion models. Numbers after layers represent the number of hidden nodes associated to the layer. See the text for more information.

followed by a fully connected layer with a size of 64 before the output layer. The activation function used is again LeakyRelu, and the outputs of the transformer layers are summed to the last layer before the first transformer block, which is observed to result in better performance according to our experiments.

Both diffusion models incorporate time information by feeding random Fourier features [51] through two fully connected layers with 32 and 64 nodes. The resulting embeddings are combined with additional conditional information including jet type for the jet diffusion model and both jet type and jet kinematic information for the particle diffusion model. After passing through a fully connected layer of size 64, these embeddings are concatenated with the inputs of each diffusion model.

A visual description of both models is shown in Fig. 2.

The implementation of the model is carried out using Keras backend [52] with a TensorFlow [53] backend. The model is trained for up to 250 epochs with a cosine learning rate schedule [54] with initial learning rate of 16×10^{-4}. If the loss function does not decrease for 20 consecutive epochs, evaluated in a separate testing set, representing 20% to the sample size, then the training is stopped. During training, 16 NVIDIA A100 GPUs are used simultaneously interfaced with the Horovod package [55] on the Perlmutter supercomputer [56]. The batch size in each GPU is set to 128. The hyperparameters used in the model architecture were optimized using the KerasTuner [57] package with Hyperband [58] algorithm.

IV. RESULTS

The performance of the generative model is evaluated using physics-based metrics proposed in [35] as well as additional metrics designed specifically to assess the quality of the jet kinematic generation. These metrics include the 1-Wasserstein (W_1) distances that are calculated using only particle information such as averaged particle relative momentum W_1^P, relative jet mass W_1^{PM}, and average of first five energy flow polynomials W_1^{PEFP} [59]. The 1-Wasserstein distances are also calculated for jet kinematic information, including jet transverse momentum W_1^{JP}, jet pseudorapidity $W_1^{J\eta}$, jet mass W_1^{JM}, and jet particle multiplicity W_1^{JN}. The evaluation also includes Fréchet ParticleNet distance (FPND), coverage (Cov), and minimum matching distance (MMD), described in Ref. [35]. To calculate each metric, 50,000 generated examples for each jet category are compared against 50,000 validation samples that were not used during training. Uncertainties are estimated using bootstrapping with replacement following [35]. Additionally, results for distilled models with a different number of total time steps are also provided in Table I.

Different jet kinematic distributions are shown in Fig. 3 as well as the comparison of the different metrics listed in Table II. We also present the per-particle distributions in Fig. 4, displaying simultaneously all particles inside a given jet. Results are also compared with the official implementations of EPiC-GAN and MP-GAN, where in the latter the MP-MP implementation is taken for the comparison.

While other implementations using the same datasets exist [37,41], these models are not directly comparable to our method as they are conditioned by the jet kinematic information, whereas our method simultaneously models both the jet and particle kinematic distributions.

Similarly, we also consider the same physics-inspired metrics to evaluate FPCD in the dataset consisting of up to 150 particles per jet. In this case, the majority of the jets used during training need to be zero-padded and is used to display the capability of FPCD to learn how to generate jets with varying number of particles. The comparison of the physics inspired metrics are listed in Table III. Since the jet kinematic information is not affected by the maximum number of particles stored in each jet, we only report the W_1^{JN} metric for each dataset in Table IV. Histograms for

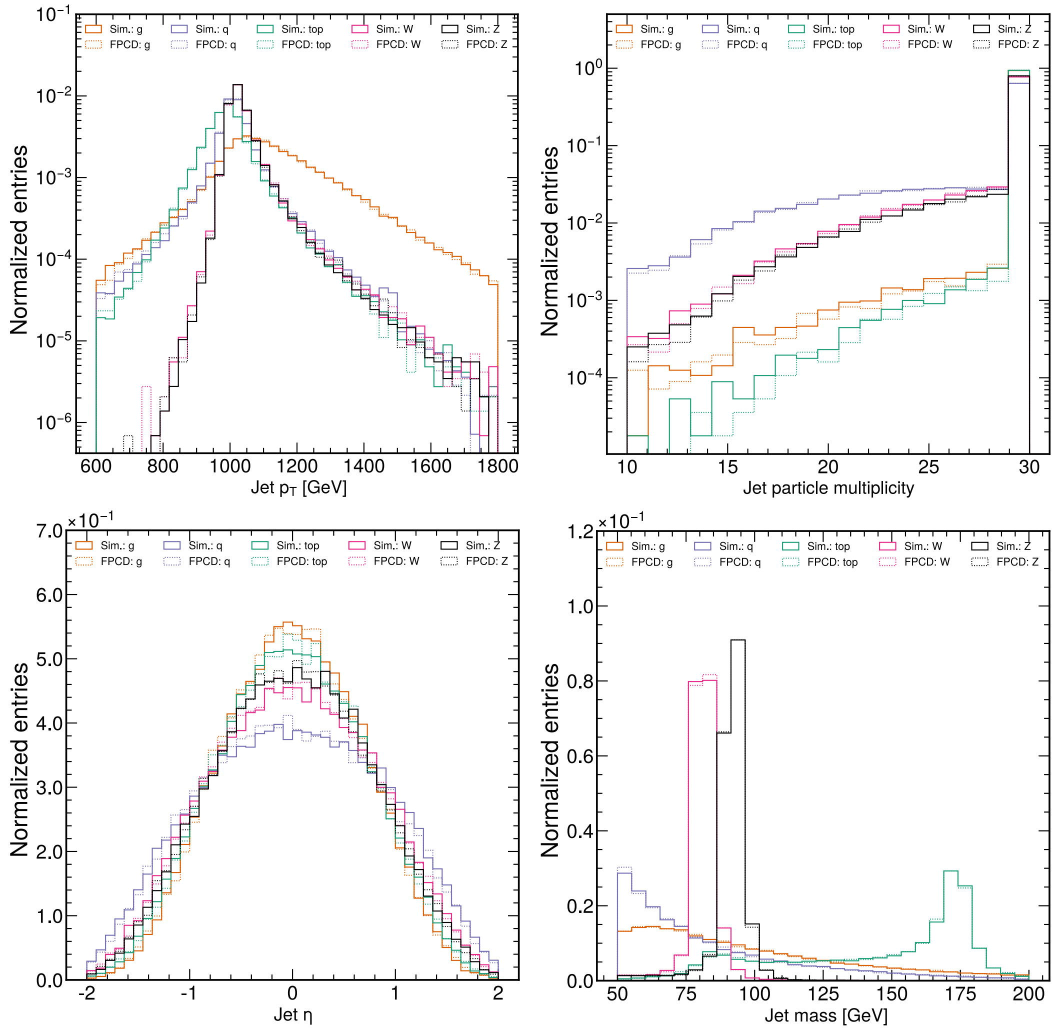

FIG. 3. Generated jet kinematic information using FPCD compared to simulated events for particle jets consisting of light quarks (q), gluons (g), and top quarks (top).

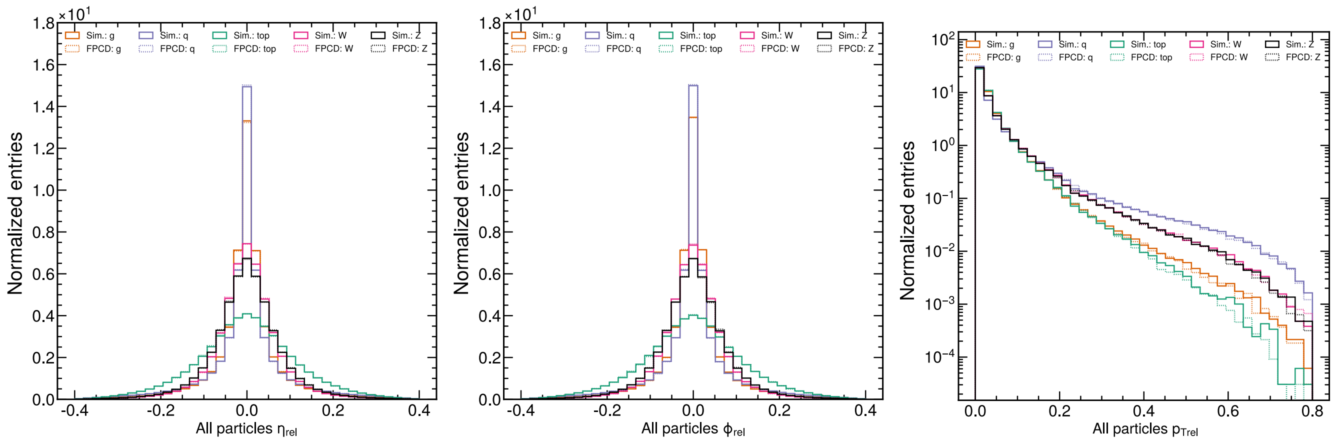

FIG. 4. Generated particle kinematic information using FPCD compared to simulated events for particle jets consisting of light quarks (q), gluons (g), and top quarks (top). For each jet, all particles for both simulation and FPCD are shown.

TABLE I. Comparison of the results obtained between different generative models in the task of particle property generation in the dataset consisting of 30 particles. Baseline FPCD uses 512 time steps during sampling. Distilled models are listed alongside number of time steps used. Lower is better for all metrics except Cov. FPND metrics are not available for W and Z bosons, hence omitted.

Jet class	Model	W_1^{PM} ($\times 10^{-3}$)	W_1^{P} ($\times 10^{-3}$)	W_1^{PEFP} ($\times 10^{-5}$)	FPND	Cov↑	MMD
Gluon	FPCD	**0.36 ± 0.08**	**0.34 ± 0.09**	**0.47 ± 0.13**	**0.07**	**0.55**	**0.03**
	FPCD 8	0.60 ± 0.16	**0.36 ± 0.07**	**0.54 ± 0.09**	**0.07**	**0.55**	**0.03**
	FPCD 1	0.65 ± 0.11	**0.34 ± 0.06**	0.60 ± 0.09	0.11	**0.55**	**0.03**
	MP-GAN [35]	0.69 ± 0.07	1.8 ± 0.2	0.9 ± 0.6	0.20	0.54	0.037
	EPiC-GAN [36]	**0.3 ± 0.1**	1.6 ± 0.2	**0.4 ± 0.2**	1.01 ± 0.07	⋯	⋯
Light quark	FPCD	**0.52 ± 0.07**	**0.27 ± 0.06**	**0.38 ± 0.11**	**0.08**	0.49	**0.02**
	FPCD 8	**0.59 ± 0.14**	0.35 ± 0.05	**0.44 ± 0.07**	0.09	0.48	**0.02**
	FPCD 1	**0.59 ± 0.08**	0.36 ± 0.08	0.50 ± 0.08	0.09	0.48	**0.02**
	MP-GAN [35]	**0.6 ± 0.2**	4.9 ± 0.5	0.7 ± 0.4	0.35	**0.50**	0.026
	EPiC-GAN [36]	**0.5 ± 0.1**	4.0 ± 0.4	0.8 ± 0.4	0.43 ± 0.03	⋯	⋯
Top quark	FPCD	**0.51 ± 0.07**	**0.41 ± 0.12**	**1.25 ± 0.19**	**0.17**	**0.58**	**0.05**
	FPCD 8	0.80 ± 0.06	**0.45 ± 0.12**	1.91 ± 0.30	0.37	**0.58**	**0.05**
	FPCD 1	1.22 ± 0.09	**0.46 ± 0.10**	2.66 ± 0.26	0.56	0.57	**0.05**
	MP-GAN [35]	**0.6 ± 0.2**	2.3 ± 0.3	2 ± 1	0.37	0.57	0.071
	EPiC-GAN [36]	**0.5 ± 0.1**	2.1 ± 0.1	1.7 ± 0.3	0.31 ± 0.037	⋯	⋯
W boson	FPCD	**0.26 ± 0.03**	**0.39 ± 0.08**	**0.15 ± 0.02**	⋯	**0.56**	**0.02**
	FPCD 8	0.48 ± 0.04	**0.38 ± 0.05**	0.22 ± 0.02	⋯	0.55	**0.02**
	FPCD 1	0.94 ± 0.06	**0.42 ± 0.09**	0.35 ± 0.03	⋯	**0.56**	**0.02**
Z boson	FPCD	**0.21 ± 0.04**	**0.40 ± 0.13**	**0.18 ± 0.03**	⋯	**0.56**	**0.02**
	FPCD 8	0.40 ± 0.04	**0.35 ± 0.04**	0.27 ± 0.03	⋯	**0.56**	**0.02**
	FPCD 1	0.99 ± 0.05	**0.35 ± 0.06**	0.49 ± 0.03	⋯	**0.56**	**0.02**

each of the distributions considered in this study are provided in Appendix A. Finally, we also compare the generation time for FPCD in Table V.

The original FPCD model is highly accurate but computationally expensive. However, we found that a distilled model with as few as eight time steps can achieve similar results while significantly reducing the overall sampling time. Surprisingly, a distilled model with only a single time step during generation still retains high fidelity while further reducing the sampling time. In Appendix B, we

TABLE II. Comparison of the results obtained between different generative models for the task of jet property generation in the dataset consisting of 30 particles. Baseline FPCD uses 512 time steps during sampling. Distilled models are listed alongside number of time steps used. Lower is better for all metrics listed except Cov.

Jet class	Model	$W_1^{Jp_T}$	$W_1^{J\eta}$	W_1^{JM}	W_1^{JN}
Gluon	FPCD	1.5 ± 0.5	0.009 ± 0.003	0.30 ± 0.07	0.020 ± 0.009
	FPCD 8	1.5 ± 0.5	0.010 ± 0.003	0.30 ± 0.07	0.021 ± 0.009
	FPCD 1	1.5 ± 0.4	0.011 ± 0.003	0.30 ± 0.10	0.025 ± 0.011
Light quark	FPCD	1.3 ± 0.4	0.008 ± 0.002	0.39 ± 0.14	0.023 ± 0.009
	FPCD 8	1.4 ± 0.3	0.009 ± 0.002	0.39 ± 0.14	0.024 ± 0.010
	FPCD 1	1.4 ± 0.3	0.009 ± 0.002	0.39 ± 0.09	0.024 ± 0.008
Top quark	FPCD	1.4 ± 0.3	0.009 ± 0.003	0.37 ± 0.12	0.022 ± 0.007
	FPCD 8	1.5 ± 0.3	0.010 ± 0.003	0.37 ± 0.12	0.023 ± 0.007
	FPCD 1	1.4 ± 0.3	0.009 ± 0.002	0.41 ± 0.12	0.025 ± 0.009
W boson	FPCD	1.19 ± 0.34	0.008 ± 0.004	0.33 ± 0.15	0.021 ± 0.010
	FPCD 8	1.23 ± 0.33	0.009 ± 0.003	0.34 ± 0.14	0.021 ± 0.010
	FPCD 1	1.21 ± 0.25	0.009 ± 0.003	0.33 ± 0.10	0.023 ± 0.011
Z boson	FPCD	1.14 ± 0.22	0.011 ± 0.004	0.34 ± 0.18	0.023 ± 0.013
	FPCD 8	1.18 ± 0.24	0.012 ± 0.004	0.35 ± 0.18	0.024 ± 0.013
	FPCD 1	1.43 ± 0.35	0.010 ± 0.004	0.36 ± 0.13	0.030 ± 0.015

TABLE III. Comparison of the results obtained between different generative models in the task of particle property generation in the dataset consisting of 150 particles. Baseline FPCD uses 512 time steps during sampling. Distilled models are listed alongside number of time steps used. Lower is better for all metrics except Cov.

Jet class	Model	W_1^{PM} ($\times10^{-3}$)	W_1^{P} ($\times10^{-3}$)	W_1^{PEFP} ($\times10^{-5}$)	Cov↑	MMD
Gluon	FPCD	**0.44 ± 0.11**	**0.28 ± 0.05**	**0.91 ± 0.16**	**0.56**	**0.03**
	FPCD 8	0.56 ± 0.06	0.40 ± 0.05	**1.09 ± 0.23**	**0.56**	**0.03**
	FPCD 1	0.65 ± 0.12	0.58 ± 0.03	1.49 ± 0.34	0.55	**0.03**
	EPiC-GAN [36]	**0.4 ± 0.1**	3.2 ± 0.2	**1.1 ± 0.7**	⋯	⋯
Light quark	FPCD	**0.46 ± 0.05**	**0.24 ± 0.02**	**0.43 ± 0.09**	**0.54**	**0.02**
	FPCD 8	**0.46 ± 0.09**	0.39 ± 0.02	0.63 ± 0.21	0.53	0.02
	FPCD 1	**0.39 ± 0.04**	0.61 ± 0.03	0.57 ± 0.10	**0.54**	**0.02**
	EPiC-GAN [36]	**0.4 ± 0.1**	3.9 ± 0.3	**0.7 ± 0.4**	⋯	⋯
Top quark	FPCD	**0.40 ± 0.07**	**0.30 ± 0.03**	**2.23 ± 0.16**	**0.58**	**0.05**
	FPCD 8	0.56 ± 0.08	0.56 ± 0.04	3.29 ± 0.11	**0.58**	**0.05**
	FPCD 1	0.85 ± 0.09	0.87 ± 0.03	3.82 ± 0.24	**0.58**	**0.05**
	EPiC-GAN [36]	0.6 ± 0.1	3.7 ± 0.3	**2.8 ± 0.7**	⋯	⋯
W boson	FPCD	**0.29 ± 0.02**	**0.23 ± 0.02**	**0.22 ± 0.04**	0.55	**0.02**
	FPCD 8	0.47 ± 0.03	0.39 ± 0.01	0.31 ± 0.04	**0.56**	**0.02**
	FPCD 1	0.93 ± 0.04	0.67 ± 0.01	0.37 ± 0.03	**0.56**	**0.02**
Z boson	FPCD	**0.28 ± 0.05**	**0.22 ± 0.03**	**0.23 ± 0.03**	0.55	**0.02**
	FPCD 8	0.52 ± 0.04	0.42 ± 0.01	0.37 ± 0.05	0.56	**0.02**
	FPCD 1	1.04 ± 0.08	0.69 ± 0.02	0.62 ± 0.06	**0.57**	**0.02**

TABLE IV. Comparison of the results obtained between different generative models for the task of jet particle multiplicity generation in the dataset consisting of 150 particles. Baseline FPCD uses 512 time steps during sampling. Distilled models are listed alongside number of time steps used.

	Model	Gluon	Light quark	Top quark	W boson	Z boson
W_1^{JN}	FPCD	0.157 ± 0.036	0.191 ± 0.067	0.171 ± 0.054	0.165 ± 0.056	0.241 ± 0.090
	FPCD 8	0.157 ± 0.035	0.190 ± 0.066	0.168 ± 0.054	0.165 ± 0.055	0.239 ± 0.090
	FPCD 1	0.157 ± 0.035	0.190 ± 0.067	0.169 ± 0.054	0.166 ± 0.055	0.239 ± 0.090

compare the histograms for each of the kinematic distributions generated by the diffusion model and the distilled models in the dataset of top quark initiated jets.

TABLE V. Timing comparison for full jet generation with FPCD. A fixed batch size of 10,000 examples is used. The time reported is the sum of the time used to generate each jet and particle kinematic information in a single GPU. Full simulation time is taken from [35].

Model	30 particles (μs)	150 particles (μs)
FPCD	5×10^3	31×10^3
FPCD 8	85	522
FPCD 1	11	66
EPiC-GAN [36]	2	12
MP-GAN [35][a]	35.7	⋯
Full Simulation	46×10^3	46×10^3

[a]The EPiC-GAN work also provides a time comparison with a retrained MP-GAN. However since we do not retrain any model used for comparison, we decided to mention only the official results released in the original publications.

EPiC-GAN is shown to be around one order of magnitude faster than FPCD even with a single diffusion step due to a different network architecture proposed by the authors. On the other hand, FPCD with a single time step is faster than MP-GAN, which also generates new jets through a single evaluation of the trained network. Nevertheless, all generative models are several orders of magnitude faster than the original physics simulation.

V. CONCLUSION AND OUTLOOK

In this work we introduced a FPCD model, providing flexibility, accuracy, and computational efficiency in jet generation. By simultaneously learning and generating multiple jet species, the two-part diffusion model generates both the jet kinematic information and particle information, conditioned on the jet kinematics and particle type to be generated.

Our model has achieved state-of-the-art performance in several physics-inspired metrics. We have demonstrated its capability to generate five different jet types with high fidelity using datasets consisting of 30 to 150 particles, showcasing the model's ability to generate jets with different particle multiplicities through a masking strategy.

Furthermore, the generation time was reduced by a factor of 450 using progressive distillation compared to the initial FPCD baseline, enabling high-fidelity generation with a single time step. This exciting result motivates future research to further reduce the model complexity and accelerate even further the sampling time.

The investigation of different backbone network designs is another promising direction to reduce generation time while maintaining high fidelity. The EPiC-GAN network structure shows great potential, with lower computational costs in higher particle multiplicity regions.

Given the flexibility of our model, we envision possible future applications in fast event generation, hadronization models conditioned on parton kinematics, and full event reconstruction conditioned on different particle types.

The code for this paper can be found at https://github.com/ViniciusMikuni/GSGM.

ACKNOWLEDGMENTS

We thank Jason Wong for thoughtful discussions. V. M., M. P., and B. N. are supported by the U.S. Department of Energy (DOE), Office of Science under Contract No. DE-AC02-05CH11231. 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 No. HEP-ERCAP0021099.

APPENDIX A: JET AND PARTICLE KINEMATIC DISTRIBUTIONS FOR THE SAMPLES CONTAINING UP TO 150 PARTICLES

In this section we show the results obtained by the diffusion model trained using the dataset consisting of jets containing up to 150 particles in Figs. 5 and 6.

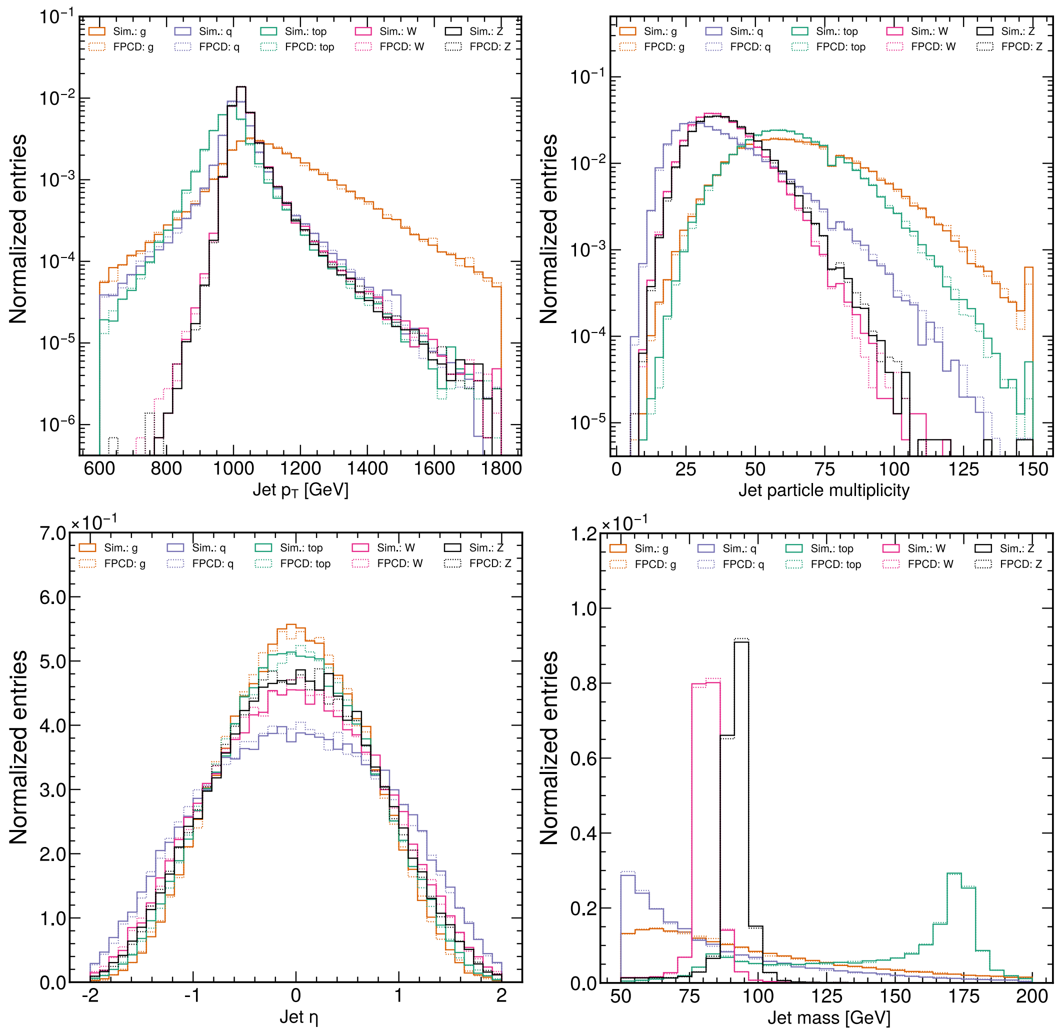

FIG. 5. Generated jet kinematic information using FPCD compared to simulated events for particle jets consisting of light quarks (q), gluons (g), and top quarks (top).

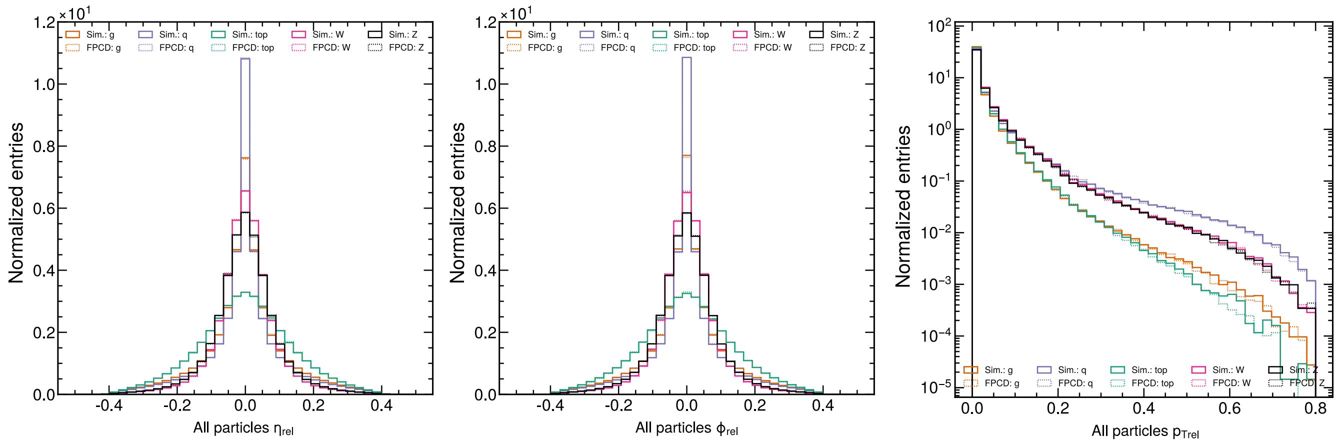

FIG. 6. Generated particle kinematic information using FPCD compared to simulated events for particle jets consisting of light quarks (q), gluons (g), and top quarks (top). For each jet, all particles for both simulation and FPCD are shown.

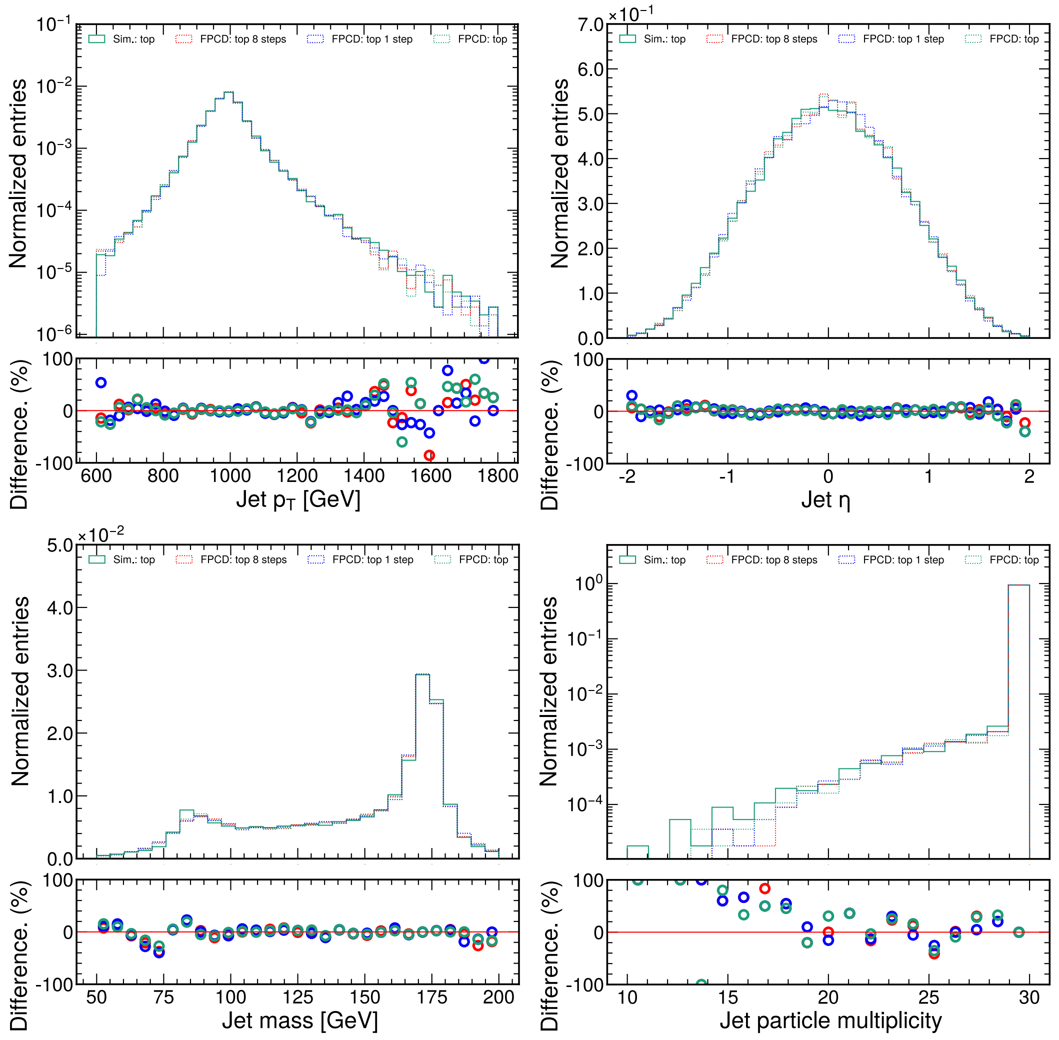

FIG. 7. Comparison of generated jet kinematic information using different distillation steps for top quark initiated jets in the dataset consisting of 30 particles.

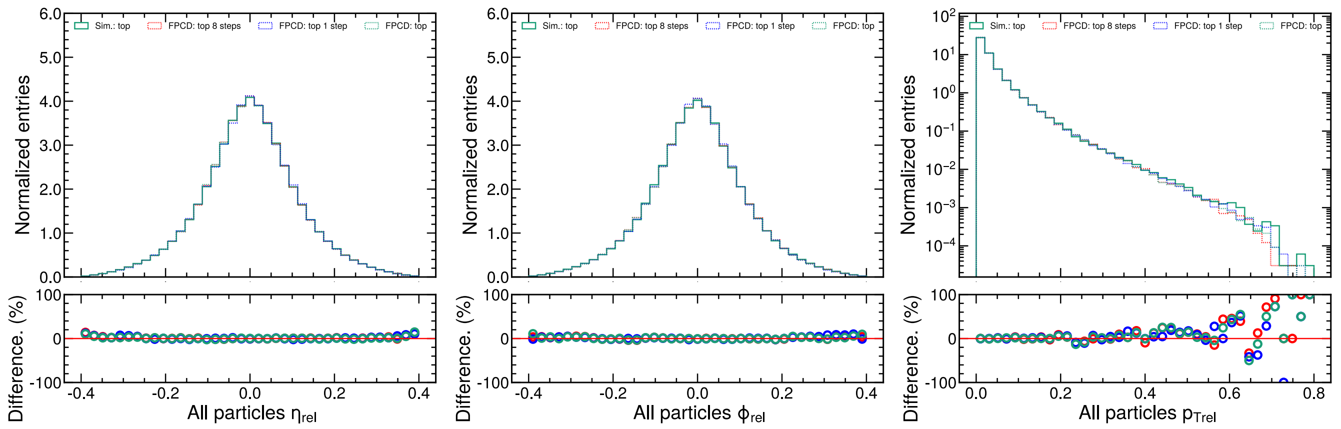

FIG. 8. Comparison of generated particle kinematic information using different distillation steps for top quark initiated jets in the dataset consisting of 30 particles.

APPENDIX B: JET KINEMATIC DISTRIBUTIONS FOR DIFFERENT DISTILLED MODELS

In this section we provide the jet and particle kinematic distributions for top quark initiated jets using different number of distillation steps. The results are shown in Figs. 7 and 8 for jet and particle information, respectively.

[1] L. de Oliveira, M. Paganini, and B. Nachman, Comput. Software Big Sci. **1**, 4 (2017).

[2] HEP ML Community, A living review of machine learning for particle physics, https://iml-wg.github.io/HEPML-LivingReview/.

[3] S. Badger *et al.*, SciPost Phys. **14**, 079 (2023).

[4] A. Butter and T. Plehn, arXiv:2008.08558.

[5] J. Cogan, M. Kagan, E. Strauss, and A. Schwartzman, J. High Energy Phys. 02 (2015) 118.

[6] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, J. High Energy Phys. 07 (2016) 069.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, arXiv: 1406.2661.

[8] J. W. Monk, J. High Energy Phys. 12 (2018) 021.

[9] M. Paganini, L. de Oliveira, and B. Nachman, Phys. Rev. D **97**, 014021 (2018).

[10] M. Paganini, L. de Oliveira, and B. Nachman, Phys. Rev. Lett. **120**, 042003 (2018).

[11] F. Carminati, A. Gheata, G. Khattak, P. Mendez Lorenzo, S. Sharan, and S. Vallecorsa, J. Phys. Conf. Ser. **1085**, 032016 (2018).

[12] V. Chekalina, E. Orlova, F. Ratnikov, D. Ulyanov, A. Ustyuzhanin, and E. Zakharov, arXiv:1812.01319.

[13] M. Erdmann, J. Glombitza, and T. Quast, Comput. Software Big Sci. **3**, 4 (2019).

[14] L. de Oliveira, M. Paganini, and B. Nachman, J. Phys. Conf. Ser. **1085**, 042017 (2018).

[15] D. Belayneh *et al.*, Eur. Phys. J. C **80**, 688 (2020).

[16] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, and K. Krüger, Comput. Software Big Sci. **5**, 13 (2021).

[17] S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman, and D. Shih, J. Instrum. **15**, P11004 (2020).

[18] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, W. Korcari, K. Krüger, P. McKeown, and L. Rustige, Mach. Learn. Sci. Tech. **3**, 025014 (2022).

[19] F. Rehm, S. Vallecorsa, K. Borras, and D. Krücker, arXiv: 2103.13698.

[20] G. R. Khattak, S. Vallecorsa, F. Carminati, and G. M. Khan, Eur. Phys. J. C **82**, 386 (2022).

[21] C. Krause and D. Shih, Phys. Rev. D **107**, 113003 (2023).

[22] C. Krause and D. Shih, Phys. Rev. D **107**, 113004 (2023).

[23] G. Aad *et al.* (ATLAS Collaboration), Comput. Software Big Sci. **6**, 7 (2022).

[24] ATLAS Collaboration, arXiv:2210.06204.

[25] S. Bieringer, A. Butter, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, B. Nachman, T. Plehn, and M. Trabs, J. Instrum. **17**, P09028 (2022).

[26] A. Rogachev and F. Ratnikov, J. Phys. Conf. Ser. **2438**, 012086 (2023).

[27] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, and K. Krüger, EPJ Web Conf. **251**, 03003 (2021).

[28] A. Abhishek, E. Drechsler, W. Fedorko, and B. Stelzer, arXiv:2210.07430.

[29] J. Liu, A. Ghosh, D. Smith, P. Baldi, and D. Whiteson, in *36th Conference on Neural Information Processing Systems* (Neural Information Processing Systems, San Diego, California, 2022), arXiv:2212.08233.

[30] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh, and D. Shih, arXiv:2302.11594.

[31] C. Krause, I. Pang, and D. Shih, arXiv:2210.14245.

[32] J. C. Cresswell, B. L. Ross, G. Loaiza-Ganem, H. Reyes-Gonzalez, M. Letizia, and A. L. Caterini, in *36th Conference on Neural Information Processing Systems* (Neural Information Processing Systems, San Diego, California, 2022), arXiv:2211.15380.

[33] V. Mikuni and B. Nachman, Phys. Rev. D **106**, 092009 (2022).

[34] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, Eur. Phys. J. C **79**, 102 (2019).

[35] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant, and D. Gunopulos, arXiv:2106.11535.

[36] E. Buhmann, G. Kasieczka, and J. Thaler, arXiv:2301.08128.

[37] B. Käch, D. Krücker, I. Melzer-Pellmann, M. Scham, S. Schnake, and A. Verney-Provatas, arXiv:2211.13630.

[38] R. Verheyen, SciPost Phys. **13**, 047 (2022).

[39] P. Dhariwal and A. Nichol, arXiv:2105.05233.

[40] T. Salimans and J. Ho, in *International Conference on Learning Representations* (2022), https://openreview.net/forum?id=TIdIXIpzhoI.

[41] M. Leigh, D. Sengupta, G. Quétant, J. A. Raine, K. Zoch, and T. Golling, arXiv:2303.05376.

[42] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, arXiv:2011.13456.

[43] J. Song, C. Meng, and S. Ermon, arXiv:2010.02502.

[44] M. Cacciari, G. P. Salam, and G. Soyez, J. High Energy Phys. 04 (2008) 063.

[45] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant, and D. Gunopulos, Jetnet (2022), https://zenodo.org/record/6975118.

[46] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant, and D. Gunopulos, Jetnet150 (2022), https://zenodo.org/record/6975117.

[47] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[48] B. Xu, N. Wang, T. Chen, and M. Li, arXiv:1505.00853.

[49] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, arXiv:1703.06114.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, arXiv:1706.03762.

[51] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, in *Advances in Neural Information Processing Systems*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Curran Associates, Inc., San Diego, California, 2020), Vol. 33, pp. 7537–7547.

[52] F. Chollet, Keras, https://github.com/fchollet/keras (2017).

[53] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, in *OSDI* (2016), Vol. 16, pp. 265–283.

[54] I. Loshchilov and F. Hutter, arXiv:1608.03983.

[55] A. Sergeev and M. D. Balso, arXiv:1802.05799.

[56] Perlmutter system, https://docs.nersc.gov/systems/perlmutter/architecture/.

[57] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, KerasTuner, https://github.com/keras-team/keras-tuner (2019).

[58] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, J. Mach. Learn. Res. **18**, 1 (2018).

[59] P. T. Komiske, E. M. Metodiev, and J. Thaler, J. High Energy Phys. 04 (2018) 013.