usualmathcalOMScmsymn usualmathcal

Choose Your Diffusion: Efficient and flexible ways to accelerate the diffusion model in fast high energy physics simulation

Cheng Jiang[1⋆], Sitian Qian[2†] and Huilin Qu[3‡]

1 School of Physics and Astronomy, University of Edinburgh, EH9 3FD, Edinburgh, United Kingdom
2 School of Physics, Peking University, 100871, Beijing, China
3 CERN, EP Department, CH-1121 Geneva 23, Switzerland

⋆ C.Jiang-19@sms.ed.ac.uk , † stqian@pku.edu.cn, ‡ huilin.qu@cern.ch

Abstract

The diffusion model has demonstrated promising results in image generation, recently becoming mainstream and representing a notable advancement for many generative modeling tasks. Prior applications of the diffusion model for both fast event and detector simulation in high energy physics have shown exceptional performance, providing a viable solution to generate sufficient statistics within a constrained computational budget in preparation for the High Luminosity LHC. However, many of these applications suffer from slow generation with large sampling steps and face challenges in finding the optimal balance between sample quality and speed. The study focuses on the latest benchmark developments in efficient ODE/SDE-based samplers, schedulers, and fast convergence training techniques. We test on the public CaloChallenge and JetNet datasets with the designs implemented on the existing architecture, the performance of the generated classes surpass previous models, achieving significant speedup via various evaluation metrics.

Received Date
Accepted Date
Published Date

Contents

1 Introduction

The Large Hadron Collider (LHC) [1] is one of the biggest particle accelerator human have ever made. More than billions of events are generated and recorded by experiments like ATLAS [2] and CMS [3] near the interaction points during each run. Sufficient and precise simulation for comparing the data and theoretical prediction is required to find the evidence of new physics at the energy frontier.

The `Geant4` software toolkit [4] is applied for the detector simulation. The simulated primary particles would propagate through specific material and geometry, resulting vast amount of secondary particles produced in each propagated steps to achieve precision. As a consequence, the detector simulations for the modern high granularity detectors occupy the most computation resources [5]. Reconstructing the physics objects like jets with complicated substructure for different physics process also plays a crucial role in providing the enough statistics for the data-driven analysis. In the upcoming High Luminosity LHC [6], the expected integrated luminosity will scale up to ten times the current level. It is nearly impossible to perform the full simulation for all the events with constrained computational budget, manifesting the necessity of fast simulation.

To address with these challenges comprehensively, different techniques adopted from deep generative modeling, where a lot of phenomenal improvements have quickly emerged over the years, have become the popular approach as fast surrogate models in high energy physics.

One latest category of generative models, the Diffusion Model (DM) as well as its variations, has become increasingly dominant in various generative applications [7–21]. There has been a few studies using DM in both fast detector and end-to-end event simulations.

For fast detector simulation, `CaloScore` [22] utilizes the score-based generative model and the conditional UNet to produce almost indistinguishable generated showers from truth ones. `CaloDiffusion` [23], on the other hand, utilizes the denoising diffusion probabilistic model, incorporates novel geometry adaptation, and enhances the UNet with middle self-attention and cylindrical convolutional layers. This combination positions `CaloDiffusion` as one of the top models in fast detector simulation. Moreover, `CaloClouds` [24] offers an alternative way to generate geometry-independent point clouds for great modelling of physically relevant distributions.

For end-to-end event simulation [20, 21], there are a series of studies for generating the point cloud with DM [25, 26]. Notably, those studies also use architectures like transformer [25] or `EPiC` layer [26] that utilize the permutation invariance of jet constituents.

Among those original studies, their first versions often suffer from a long generation time, as desired results typically converge only after a large number of sampling steps. Further studies involving the fresh distillation methods like progressive distillation [27, 28], consistency

model [23, 29, 30] to get the result in only few steps with the exchange of longer training time, and other ways [29, 31] to speedup the backward process.

Most of the time, different DM methods face a tradeoff between the quality and speed of generated classes. There is a lack of a comprehensive study on the training and sampling dynamics of diffusion models applied for various purposes and datasets in high energy physics. In this study, we investigate the effects of different training strategy by offering a soft way to expedite the convergence of performance. We also adopt several popular training-free ODE/SDE based samplers and noise schedulers in post-DDPM era [32–36]. By balancing the discretization errors and accumulated errors in the sampling process, a flexible and efficient way can be offered to make the diffusion model faster and more accurate.

We finally make our approaches portable by testing in different network and different datasets using the public fast and high-fidelity calorimeter dataset *CaloChallenge* [37, 38] and the jet point cloud dataset *JetNet* [39]. The performance surpasses the previous model by several evaluation metrics [8, 25, 37, 38, 40].

This paper is outlined as follows, Section 2 covers the fundamentals of diffusion model, elucidating how data such as shower images or point clouds are specifically generated through the backward process. Section 3 provides the a brief introduction about the dataset used in this study. Section 4 discusses the training strategy applied. Various metrics will be presented to evaluate the performance in Section 5. Lastly, Section 6 gives a empirical discussion, conclusion and outlooks about the study.

2 Dynamics in the Diffusion Model

The diffusion model was initially introduced to sample complex data distributions by drawing inspiration from concepts in non-equilibrium thermodynamics [41]. At first, we gradually add noises to an initial data sample from $\boldsymbol{x_0}$ to $\boldsymbol{x_T}$ until the time step $\boldsymbol{T}$, usually denoted as the forward process $\boldsymbol{q}$ where the data points are degraded as $\boldsymbol{T}$ increase. Then it follows a backward process $\boldsymbol{p}$ where noisy data could be denoised iteratively to new observation $\sim \boldsymbol{x_0}$ as the time goes reversely.

Two main classes are introduced later [42, 43]: Denoising Diffusion Probabilistic Models (DDPMs) [44, 45]and Score-based Generative Models (SGMs) [46].

For DDPM, the forward process transforms input distribution into a prior distribution by

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t|\sqrt{1-\alpha_t}x_{t-1}, \alpha_t) \tag{1}$$

where $\boldsymbol{\alpha_t}$ is the noise variance in every time step in case of the noise has unit variance $\boldsymbol{\epsilon \sim \mathcal{N}(0, \mathcal{I})}$, the single step formulation can be rewritten as:

$$q(x_t|x_0) = \mathcal{N}(x_t|\sqrt{\tilde{\alpha}_t}x_0, 1-\tilde{\alpha}_t) \tag{2}$$

$$x_t = \sqrt{\tilde{\alpha}_t}x_0 + \sqrt{1-\tilde{\alpha}_t}\epsilon \tag{3}$$

where $\tilde{\alpha}_t = \prod_{i=1}^{T}(1-\alpha_i)$. The direct calculation from $\boldsymbol{x_0}$ can allow us to random sample time steps to add noise during training. In the backward process, a probability function $\boldsymbol{p(x_{t-1}|x_t)}$ is learned to denoise the noisy data at previous time step until reach the initial input $\boldsymbol{x_0}$.

The SGM is formularize similarly, in the score Stochastic Differential Equations (SDEs), the forward process is defined by [46]:

$$dx = f(x,t)dt + g(t)w_t \tag{4}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are drift and diffusion coefficients chosen differently for variance preserving and variance exploding noise, $\boldsymbol{w}$ is the standard Brownian motion.

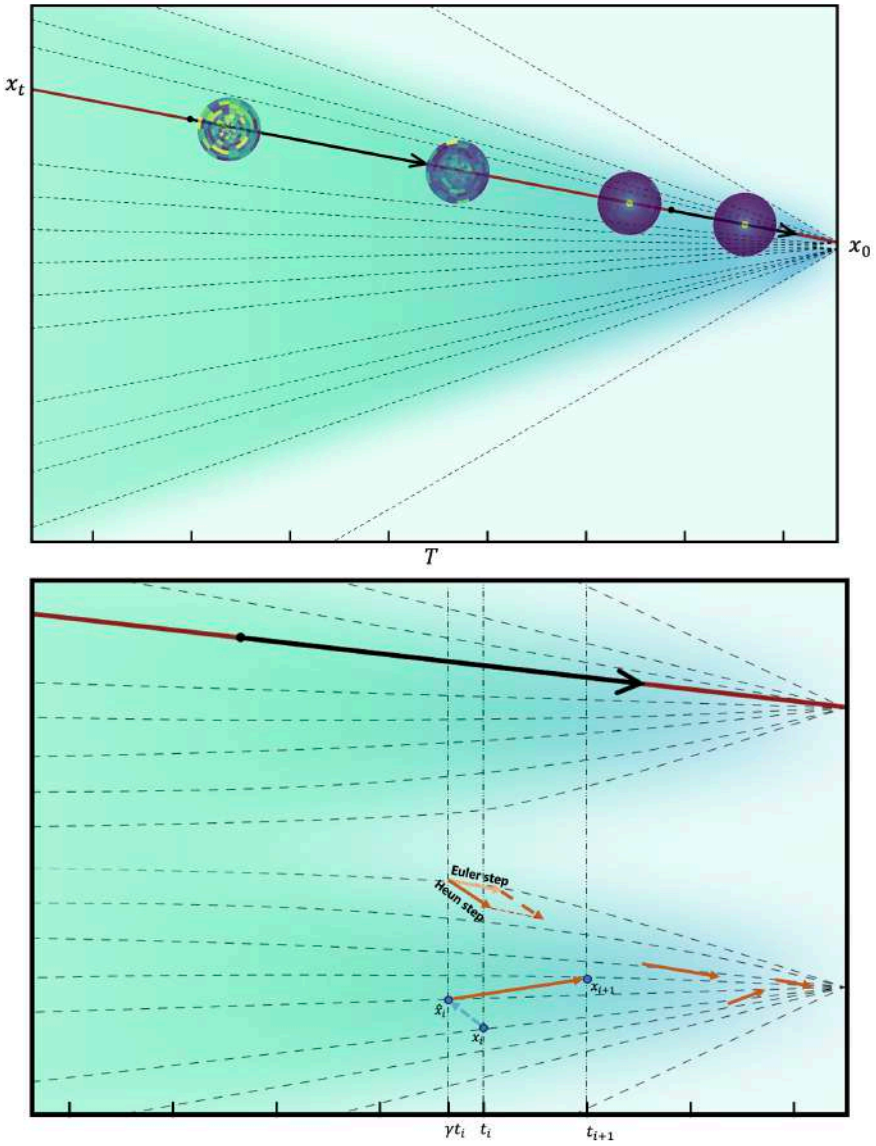

Figure 1: Upper: The schematic diagram for a noisy shower image (energy deposited in one layer of calorimeter) $\boldsymbol{x_t}$ denoises along the ODE curvature with varying time steps to clean shower close to $\boldsymbol{x_0}$. Lower: Arrows indicate a schematic sketch how ODE and SDE ~~ways to get~~ samplers reach the $\boldsymbol{x_0}$ along timesteps, ODE flows along the certain direction toward $\boldsymbol{x_0}$. $\boldsymbol{t_i}$ is the $\boldsymbol{i^{th}}$ timestep, ~~where~~ SDE will have extra noise injection $\boldsymbol{\gamma t_i}$ in each time step before ~~doing the ODE part~~moving to next step. The Euler and Heun are the first and second order calculation for updating $\boldsymbol{x_i}$.

The reverse-time SDE is defined as:

$$dx = [f(x,t) - g(t)^2 \nabla_x \log p_t(x)]dt + g(t)d\tilde{w} \tag{5}$$

unlike DDPM, the quantity we are solving is the score function (A vector field indicating the direction toward regions of increased data density given a certain level of noise.) $\nabla_x \log p_t(x)$. The learning process of this gradient of log probability functions offers another possibility to use not only SDE but also ODE to get the solution.

Then follows by the generalization from [47], we can write the reverse SDE equation as a sum of probability flow ODE and Langevin diffusion SDE:

$$dx = -\dot{\sigma}(t)\sigma(t)\nabla_x \log p_t(x)dt - \beta(t)\sigma(t)^2 \nabla_x \log p_t(x)dt + \sqrt{2\beta(t)}\sigma(t)dw_t \tag{6}$$

where $\sigma(t)$ is the noise schedule, $\beta(t) = \dot{\sigma}(t)/\sigma(t)$, the first term is the marginally equivalent ODE, the second and third terms are added as the deterministic flow and noise injection term for Langevin diffusion, the schematic diagram as shown in Fig. 1.

This determines the characteristics of the general ODE/SDE solver. ODE solvers often have smaller discretization errors than SDE solvers in smaller sampling steps [49,51]. However, as the sampling steps become larger, the accumulated error for SDEs would be much smaller than that for ODEs, resulting in a smaller total error (better performance) in large sampling steps. The noise scheduler determines how fast those samplers should learn along different time steps. In the paper, we will cover up to seven mainstream samplers nowadays (four ODEs and three SDEs) over different schedulers.

The schedulers and samplers directly decide how the noisy data x_t will reach x_0 with proper noise levels. Our choices for schedulers and samplers are listed belows:

- Schedulers:
  - The default cosine beta scheduler for discrete time DDPM pretrained model, this is the baseline to compare with performance for other schedulers and samplers. $\bar{\alpha} = \cos(\frac{\pi(t/T+s)}{2(1+s)})$ where $s = 0.008$ that determines the minimum noise for last steps. We can tell by the function, the noises are added by a linearly decrease rate during intermediate steps to avoid too much noise in the last few steps.
  - Variance Preserving schedulers ~~used in~~ as a more generalized version of cosine beta schedulers used in both score matching [46] and DDPM, defined as a continous time step t and $\sigma = \sqrt{(\exp\frac{\beta_{\max} t^2}{2} + \beta_{\min} t - 1)}$ where $\beta_{\min}$ and $\beta_{\max}$ are constant to determine the lower and upper bound of noise level.
  - The schedulers proposed by Karras [47]. ~~$t_{i<N} = (\sigma_{\max}^{1/\rho} + \frac{i}{N-1}(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}))^\rho$~~ $\sigma_{i<N} = (\sigma_{\max}^{1/\rho} + \frac{i}{N-1}(\sigma$ and $\sigma_N = 0$. It's the monotonically decreasing function, so the noise added in a decreasing rate, ρ determines the curvature. Larger ρ means more steps near $\sigma_{\min}$ are shortened. Reasonable value for $\sigma_{\min}, \sigma_{\max}, \rho$ in our study should around $[0.01, 0.001], [20, 80], [5, 10]$ respectively. This scheduler is helpful for generating good quality of data in low sampling step region.
  - Combine Karras schedulers proposed with Lu [52] by transforming the σ to log scale. ~~$t_{i<N} = (\log(\sigma_{\max})^{1/\rho} + \frac{i}{N-1}(\log(\sigma_{\min})^{1/\rho} - \log(\sigma_{\max})^{1/\rho}))^\rho$~~ $\sigma_{i<N} = (\log(\sigma_{\max})^{1/\rho} + \frac{i}{N-1}(\log$ The values for $\sigma_{\min}, \sigma_{\max}$ remains the same region, $\rho = 1$ to keep a healthy noise level range. This scheduler is even more decreasing than Karras schedulers, so the step size changes more rapidly in low sampling step while more slowly in last steps.
- Samplers:

- The same DDPM samplers are used in [23, 44]. The Euler-Maruyama method approximates the reverse diffusion process by iteratively denoising the sample while adding stochastic noise at each discrete timestep.
- EDM Stochastic sampler with Heun's method [47], as illustrated in Fig. 1, a noise injection step is performed to current time step $\hat{t}_i = t_i + \gamma t_i$ where γ is determined by three parameters $\gamma = \min(S_{churn}/N, \sqrt{2}-1)$ when $S_{\min} \leq t_i \leq S_{\max}$. The $S_{\max}, S_{\min}$ is the range for injection, S_{churn} is the stochasticity strength which give the lower bound to start adding noise. It controls how fast ~~it~~ the sampler converges to zero noise. The optimal sets of ~~this~~ parameters can make the sampling faster and better. Contrarily, a bad sets of those stochastic term will make possible degradation of the generated sample. In our study, such degradation only happens when extreme values (e.g. large γ, or S_{churn} close to initial timestep) are selected. The difference between this approach and Euler Maruyama is that the denoising steps in EDM samplers happens in some intermediate state $\hat{t}_i$ instead of actual timestep state t, the difference between two states become diminishing when N goes large. This SDE sampler, unlike other SDEs that only give satisfied result after a very large sampling steps, can converge quickly ~~becuase of this~~if one can manage to tune those stochastic parameters well.
- Restart sampler proposed in [49]. This sampler introduces a way to strongly contract the accumulated error based on EDM ODE Heun sampler. A back-and-forth ODE step will repeat K times in the predefined single or multiple intervals $[t_{\max}, t_{\min}]$.
- DPM-Solver samplers proposed in [51,52]. A popular ODE solver that can speedup the sampling process by applying change-of-variable to noise level. We choose to use the latest multistep solver that calculate iteratively by the knowledge of current and next timestep.
- Combined DPM-Solver and EDM SDE samplers. Instead of using Euler and Heun step, it uses the "mid-point" step by change-of-variable to intermediate state $\hat{t}_i$.
- LMS sampler [53]. For each iteration, the sampler will denoise the input data using the model and updates the x_i using the linear multistep integration method with the calculated coefficients of predefined orders.
- Uni-PC sampler introduced in [54]. A unified predictor-corrector framework that can be applied on many existing samplers. We apply it with DPM-Multistep-sampler with both varying coefficient and $B(h)$ solver proposed in the paper.

3 Dataset

3.1 CaloChallenge

Fast Calorimeter Simulation Challenge 2022 [37] is the GEANT4 simulation for particle showers deposited in the calorimeter. The purpose for this challenge is to develop and benchmark the fast and high-fidelity calorimeter simulation using modern techniques from deep learning. Three datasets are provided ranging from easy to hard. The complexity of dataset is determined by the dimensionality of the calorimeter showers, including factors such as the number of layers and voxels on each layer.

In each dataset, particles are simulated to propagate along the z-axis inside the cylindrical and concentric detector. Similar to the sampling layers in the common calorimeter, a discrete number of layers are placed with specific radial and angular bins r and α. Two classes of quantities are stored: the incident particle energies and voxel energies stored in each shower.

By mapping the voxel location with given r and α bins, one can reconstruct the layer deposited energy inside the shower. This offers flexibility for designing the generative modelling tasks using either low or high level features.

This study will primarily focus on testing dataset 2 which simulates a physical detector with 45 layers of absorber material (Tungsten) and active material (Silicon). A total of 100,000 electrons in each file enter the calorimeter with a direction. The incoming particle energies span over a log-uniform distribution from 1 GeV and 1 TeV. Each layer contains 16 angular and 9 radial bins, so 16×9 = 144 cells, and a total number of 144×45 = 6480 voxels in each shower. A minimal cutoff around 15 keV is applied to voxel energy. [1]

3.2 JetNet

JetNet [39] is the point cloud like dataset which consists of 200,000 simulated events. Each event contains the corresponding jet types with transverse momenta $p_T \sim 1$ TeV, originating from either gluons, top or light quarks. Additionally, a narrow kinematic window is imposed, ensuring that the relative transverse momenta of the generated parton and gauge boson $\Delta p_T/p_T \leq 0.1$. The jet clustering is performed using the anti$-k_T$ algorithm [55] within a cone size of $R = 0.4$.

For up to 30 particle constituents stored in one jets by descending p_T order, three features are provided: the relative angular coordinate of particles w.r.t. the jets: $\eta^{rel} = \eta^{particle} - \eta^{jet}$, $\mathbf{mod}\,(\phi^{rel} = \phi^{particle} - \phi^{jet}, 2\pi)$ and relative transverse momentum $p_T^{rel} = p_T^{particle}/p_T^{jet}$. There is also a fourth variable that masks whether the particles is genuine or zero-padded when there are fewer than 30 constituents stored in the jets.

We will show preliminary results for testing on gluon and top quark jets. The gluon jet can serve as a baseline test since it usually produces the dominant background for many physics processes and has a relatively simple topology. The top jet has more subtle substructure and presents challenging tasks for generative models.

4 Preprocessing and Training

In this section, we provide an overview for the data processing, and training details on two datasets mentioned in Section 3.

4.1 CaloChallenge

We follow a common preprocessing procedure for *CaloChallenge* dataset.

- First, the voxel energies v_i are normalized by the incident particle energy.
- Then a logit-norm transformation is applied to ensure input data has zero mean and unit variance.

$$\begin{aligned} u_i &= \log\left(\frac{x}{1-x}\right), x = v_i - 2\delta v_i + \delta \\ u_i' &= \frac{u_i - \bar{u}}{\sigma_u} \end{aligned} \tag{7}$$

  Here, $\delta = 10^{-6}$ is introduced to avoid the discontinuity.

[1] The rationale behind selecting this dataset is that previous studies indicate it exhibits a comparable level of difficulty to the more finely granulated dataset 3, all while achieving training and inference speeds that are ten times faster.

We choose to use the network from [23] which is composed of a conditional UNet with middle layer self-attention layer, cynlindircal convolution and Geometry Latent Mapping (GLaM). The latter two techniques are found to be beneficial for generating quantities in r and α bins. In order to demostrate the portability of our studies, the neural network architect remains unchanged from the version in the literature [23]. The original training is tailored for discrete noise level samplers like DDPM [44].We have integrated our training setups, including different loss function designs and data preprocessing, into the new training process. We use publicly available pre-trained models with 400 DDPM steps as the baseline to compare the performance.

4.2 JetNet

For *JetNet* dataset, we follow the preprocessing procedure from [25,26]. The Equivariant Point Cloud (EPiC) layer based on the deep sets framework is chosen to be used as it shows great quality of generated classes while having lighter architecture than transformer-based models [26]. With the same training procedure changes, we further modify the architectures with similar sinusoidal and cosine positional embedding by leveraging sine and cosine functions of the noise levels.

Training the diffusion model is essential for generating higher quality showers across different noise levels. We delve into two questions: 1. How can the data with added noise be correctly weighted and learned by the model so it can be denoised more easily in the backward process? 2. How can the training converges faster to a better reliable result?

4.3 Training Details

We follow the way from [47] to rewrite the score function to a denoiser function which minimizing the ~~L2 denoising error~~ L^2 denoising error between x and model output $D(x)$ at each noise level:

$$\nabla_x \log p_t(x) \log p_\sigma(x) = (D(x) - x)/t\sigma^2 \tag{8}$$

If the network is preconditioned with σ independent skip connections, D can be written as $D_\theta(x,\sigma) = c_{\text{skip}}(\sigma)x + c_{out}(\sigma)F_\theta(c_{in}(\sigma)x, \sigma, z)$. A set of preconditioning parameters used to predict noise ϵ with this denoiser function:

$$\mathscr{L} = \mathbb{E}_{t,\epsilon}[w(t)||F_\theta(c_{in} * x_t, t, z) - \frac{1}{c_{out}}(x_0 - c_{\text{skip}}(t) * (x_t))||_2^2] \tag{9}$$

$$\begin{aligned} &_{\sigma,\epsilon}[w(\sigma)||D_\theta(x,\sigma) - x_0||_2^2] \\ &= \mathbb{E}_{\sigma,\epsilon}[w(\sigma)||F_\theta(c_{in} * x_\sigma, \sigma, z) - \frac{1}{c_{out}}(x_0 - c_{\text{skip}}(\sigma) * (x_\sigma))||_2^2] \end{aligned} \tag{10}$$

where F_θ is the preconditioned network, c_{in}, c_{out} maps the input and output data magnitude, and c_{skip} controls the skip connection of the network. ~~$c_{in} = 1/\sqrt{t^2+\sigma_c^2}$ and $c_{out} = (t * \sigma_c)/\sqrt{t^2+\sigma_c^2}$,~~ The parameters $c_{in} = 1/\sqrt{\sigma^2+\sigma_c^2}$ and $c_{out} = (\sigma * \sigma_c)/\sqrt{t^2+\sigma_c^2}$, σ is the current noise levels, σ_c is the constant value, a reasonable value is around $[0.5, 1.5]$ to avoid large variation for gradient scale. ~~The $c_{\text{skip}} = \sigma_c^2/(t^2+\sigma_c^2)$~~ $c_{\text{skip}} = \sigma_c^2/(\sigma^2+\sigma_c^2)$ is important quantity to partially mask the skip connection when the noise level goes large. This can prevent the network from inaccurate prediction. The setting is important in our case given noise varies widely in most of samplers we show later. When the ~~t~~ σ goes higher, the c_{skip} becomes smaller. In

a way, this compels the skip connection to be deactivated, encouraging the network to prioritize learning the signal for overriding the input rather than predicting noise to correct it. We choose the continuous timestep training where $\log t = \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$ where $P_{\text{mean}}, P_{\text{std}}$ is -1.2 and 1.2 correspondingly.

From our study, the choice of the weighting function is very important as it not only determines the speed of loss convergence but also the quality of the outputs and how the specific energy distribution tends to evolve across different time steps. To mitigate the instability of the optimization over the training, and balance the gradient magnitude over diverse noise levels, one loss weight category is often mentioned — Signal to noise ratio (SNR) weighting [32, 56–58]. We choose the min-SNR weighting as it is found to be beneficial for training high-resolutions dataset. The original min-SNR weighting proposed from [59] defined to be min{SNR(t),γ}, where SNR(t) is defined as $\frac{1-\sigma^2}{\sigma^2}$, γ is the upper limit for weighting value, usually set to be 2-5. This could avoid the model putting too much weight on the low noise level.

In stead of directly clamping a constant value, we propose a softer version of min-SNR weighting which is more flexible to find the optimal value for better training. Specifically, the weight function is

$$w(t) = (t * \sigma * \sigma_c)^2 / (t\sigma^2 + \sigma_c^k)^2 \tag{11}$$

that ~~t~~ σ and σ_c ~~is~~ are the same as the preconditioning parameters, the reasonable value of k in the range of $[2, 4]$. This function ~~is even with~~ has one minimum value of 0 around zero t and two local maxima on positive and negative t-axes. The constant k determines the local maximum weight value, and the SNR value where the maxima start to smoothly fall down to 0. The advantages of this weighting scheme will be shown later in Sec 5 with comparison of default constant (unit) weighting strategy.

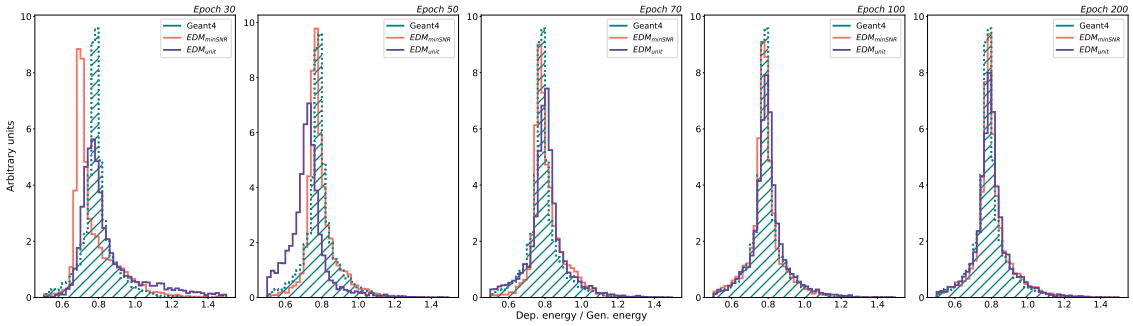

Figure 2: The distribution for total shower energy divided by the incident particle energy, tale ~~dotted line~~ shaded region is the GEANT4 reference, ~~red~~ the solid line is the generated distribution by 79 steps EDM samplers ~~. Upper:~~ with different weight at 30, 50, 70, 100, 200 epochs. Purple: default denoiser training with unit weight. ~~Lower~~Pink: ~~30, 70, 100, 200 epochs training with our~~ proposed min-SNR weighting. The default training still struggles to learn this variable after 200 epochs whereas the training with proposed weighting strategy already gives good shape alignment after 100 epochs.

5 Performance

~~We have previously discussed various~~ Various schemes for the backward diffusion process ~~. In order to~~ have been discussed in previous section. To comprehensively compare their perfor-

mance and enhance understanding of ~~how to choose~~ choosing the appropriate samplers and schedulers for a specific purpose, we offer several methods for evaluation.

5.1 CaloChallenge

There are multiple important distributions related to the layer and voxel energies for *CaloChallenge* dataset. We compare the generated samples with GEANT4 as reference. The voxel energy distribution is defined as the distribution of raw generated low level voxel energies. The total energy distribution is defined as the distribution of total energy of the generated shower. The center energy of the shower is defined as $\bar{x} = \frac{\langle x_i E_i \rangle}{\sum E_i}$ for cell location x_i and energy E_i, and shower width is defined as $\sqrt{\bar{x^2} - \bar{x}^2}$, we will show these two quantities at the polar coordinate α and r. The mean layer energy distribution defined as the mean of generated layer energies. The shower response E_{ratio} ~~defined as~~ is the total shower energy divided by the incident particle energies, this quantity is extremely important ~~as a correction factor~~ for energy calibration of many particles at LHC. As it will be used to correct the deposited shower energy inside calorimeter into the real particle energy, the small difference between generated sample and GEANT4 means more consistency for particle reconstruction procedure of both fast and full simulation. We will also evaluate the separation power between generated E_{ratio} and reference one.

In addition, a public evaluation network [37] DNN classifier with 2 layers, each comprising 2048 nodes and a 0.2 dropout rate, is employed to assess the area under the receiver operating characteristic curve (AUC) for all high-level features, including center energies, shower width, and layer energies, of both generated samples and references. An AUC value that close to 0.5 suggests that the generated and target classes are nearly indistinguishable. Conversely, a higher AUC value indicates a more evident distinction between the quantities of the two classes.

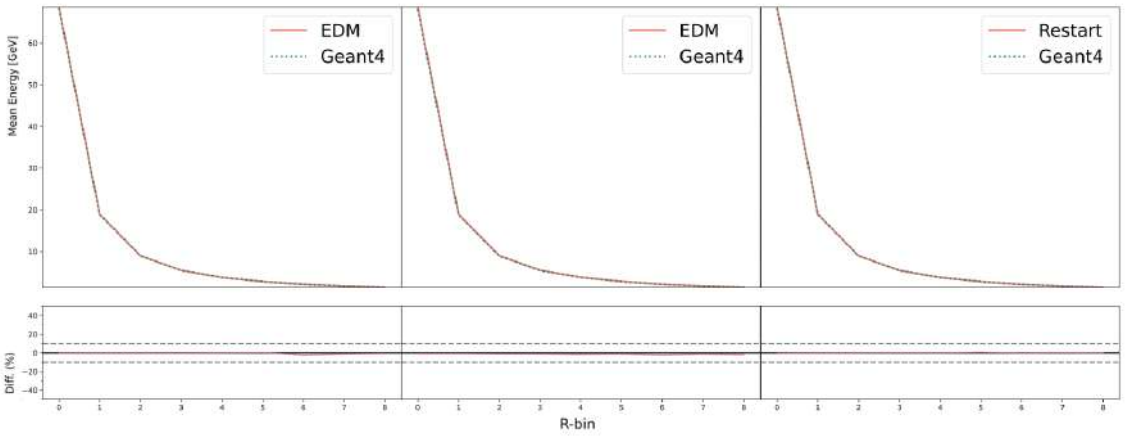

Figure 3: Comparison plots for average center energy of 5000 showers in radial bins. left: EDM with 30 steps, middle: EDM with 79 steps, right: Restart with 30 steps. Those samplers are chosen because of their good sample quality and inference speed trade-off, while the difference almost indistinguishable.

First, a comparison ~~plots~~ plot for E_{ratio} with default EDM training loss weight and our proposed training weight is shown in Fig. 2. ~~Several observations can be made from the plots.~~ The choice of the weighting function influences how the kinematics distribution evolves over time. After 200 training epochs, the model trained with the default weighting still struggles to learn the E_{ratio} distribution. In contrast, employing proposed min-SNR weighting results in

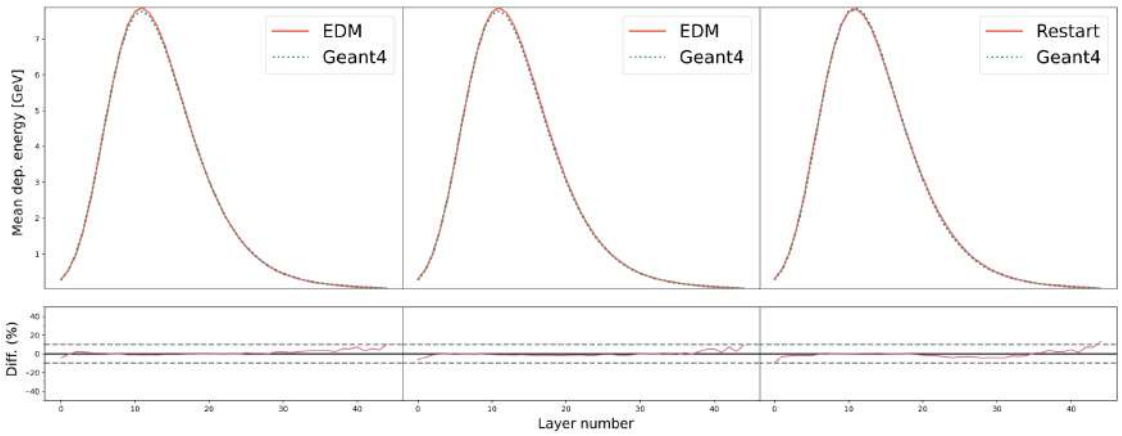

Figure 4: Comparison plots for mean layer energies of 5000 showers. left: EDM with 30 steps, middle: EDM with 79 steps, right: Restart with 30 steps.

a better alignments with reference histogram, the generated $\boldsymbol{E}_{\text{ratio}}$ aligning with ~~the reference shape~~ GEANT4 sample after only 100 epochs. This observation affirms that min-SNR weighting expedites the convergence of high-level features to achieve better performance compared to constant weighting. [2]

Few plots for center energy in $\boldsymbol{r}$ direction are selected in Fig. 3. We choose the EDM Heun SDE and Restart samplers that have good sample quality and inference speed trade-off. Notably, the EDM samplers and Restart samplers with Karras schedulers can learn this quantities really well. The ratio difference is almost indistinguishable with naked eyes. ~~We can tell from the~~ The default scheduler for the EDM sampler is Karras. To better illustrate the impact of different noise schedulers on the same sampler, a broader exploration of scheduler choices—such as Karras, DPM++ , and Lu schedulers with varying decay rates for $\boldsymbol{\sigma}$ is conducted. For small ($\sim$30) steps EDM, the differences in last three radial bins are larger meaning it underestimates the lower energy region. When the steps become larger, we see the EDM predicts well in all radial bins. The Restart sampler can achieve this with only 30 steps. ~~Because~~ Given the Restart samplers works with a predefined parameters for configuring the EDM ODE~~. The~~ , the generation time is directly proportional with the restart iterations in the configuration, usually longer than default EDM samplers. A lack of principled way for these hyperparameters is the inherent limitation for this ~~samplers~~sampler. Later we will see the best performance is achieved around 30-70 steps as we only ~~uses~~ use specific configurations in this range.

Then the mean layer energies, total shower energies, shower response, and voxel energies for the same three cases are shown in Fig. 4, 5, 6~~(right)~~.

~~Most of the samplers need a relatively long sampling time to learn the mean energy of the last few layer correctly since they have voxels with lowest energy thresholds.~~ The low steps EDM sampler has good performance on low energy voxel while mediocre performance on high energy tails of voxel distributions~~and low energy shower~~ . In Fig. 5, we have seen the shape of shower response between generated and reference sample for Restart and EDM samplers are more aligned than other samplers, especially around the peak of histogram ($\sim$0.8). DPM-solver ~~samplers~~ sampler with Karras scheduler does very well on predicting the voxel energy around 15 steps, achieving an exceptional agreement on both low and high energy tails shown in the left plot of Fig. 6. ~~All proposed samplershave comparable and better performance with smaller steps. Indeed~~ The energy thresholds also affect low voxel energies generated by dif-

[2] A 1D layer diffusion can generate better $\boldsymbol{E}_{\text{ratio}}$ distribution. But it might be over simplified as the model only considers the layer deposited energy. The paper only considers the low level voxel diffusion as a more realistic and challenging case for fast calorimeter simulation

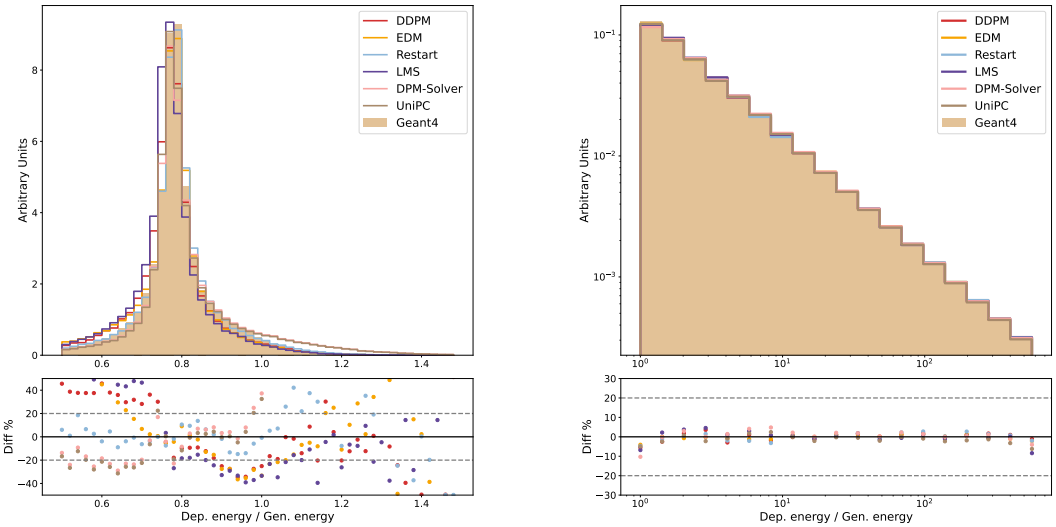

Figure 5: Comparison ~~plots~~ histograms for ~~mean layer~~ 100k shower energies~~of 5000 showers~~. left: ~~EDM with 30 steps~~shower response, ~~middle~~right: ~~EDM with~~ total shower energy from 400 steps DDPM (CaloDiffusion), 79 steps EDM, ~~right: Restart with~~ 30 steps ~~.~~Restart 36 steps LMS, 15 steps DPM-Solver ++, and 15 steps UniPC samplers

~~Comparison histograms for 5000 total shower energies. left: EDM with 30 steps, middle: EDM with 79 steps, right: Restart with 30 steps.~~

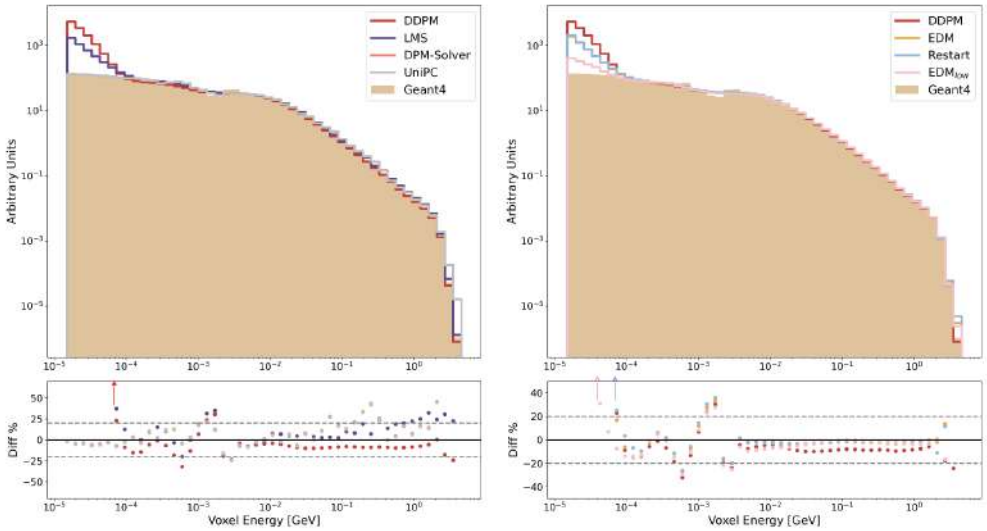

Figure 6: Comparison histograms for raw voxel energies of ~~5000~~ 100k showers: Left: 30 steps EDM, 79 steps EDM, 30 steps Restart samplers; Right: 400 steps DDPM (CaloDiffusion), 36 steps LMS, and 15 steps DPM-Solver ++ samplers. Bottom is the percentage of difference between Geant4 and generated samples. The upward arrows indicate that the differences fall beyond the window at certain voxel energy levels.

ferent samplers. For the baseline comparison, we used the pre-trained CaloDiffusion [23] model for dataset 2 with default 400 steps DDPM has 4-10% difference in the intermediate energy level (5-1000 MeV), and less than 30% difference with GEANT4 until 0.7 MeV. A clear difference has been seen below 0.7 MeV. The same scenario happens to EDM, Restart and LMS samplers, but all three samplers show more robust performance (i.e. a smaller excess in low energy tail). Generated showers produced by EDM and Restart samplers both have much less difference with GEANT4 around the intermediate energy level. Two new ODE samplers, DPM-Solver and UniPC, hold only less than 10% difference in low energy region, nevertheless perform worse on other regions. Indeed, some of them are struggling to match the low voxel energy, the presence of the tail is probably a consequence of model itself and too low energy threshold. It is expected this deficiency should not have big impact on the downstream reconstruction of shower.

~~However,~~

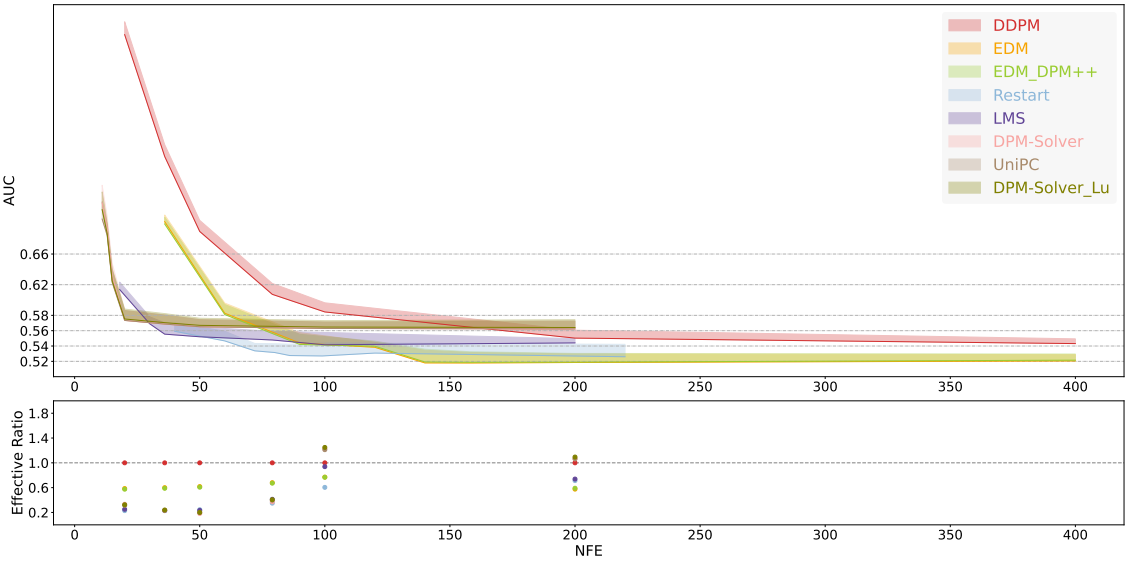

Figure 7: AUC values for different samplers as a function of evaluation steps, the shaded area is possible AUC value range for 20 evaluations through independent classifier training. The effective ratio is defined as the ratio between actual AUC from generated sample with different solvers minus the best possible AUC value 0.5, divided by the AUC value from generated sample with DDPM minus 0.5 $\frac{\text{AUC}_{gen}-0.5}{\text{AUC}_{DDPM}-0.5}$ Lower is better, value below 1 means better performance than DDPM at certain evaluation steps.

We showed the possible AUC values for high level feature classifiers with different samplers and schedulers by evaluating different samplers 20 times in Fig. 7. DPM-solver sampler seems to perform poorly on predicting the low and high-level features simultaneously. The Uni-PC sampler has very similar performance with DPM-solver sampler and only gives slightly better performance over very low sampling (~~5-10~~10-20) steps. The lowest AUC ~~for high level feature classifiers is around 60 at 20~~is around 57 at 50$^{\text{th}}$ step. ~~We showed the AUC plots for evaluating different samplers 20 times in Fig. 7.~~ The EDM and Restart samplers ~~surpasses~~ surpass the previous 400 steps DDPM samplers at 50, 25 steps (100, 60 function evaluations) correspondingly, combined with other plots we showed, achieving the same and better performance with a 3-8 times speedup. The LMS ODE sampler has a better performance than EDM below 45 steps, but reaches a worse convergence than other samplers. The best AUC value for Restart samplers ~~achieves~~ achieve at 30-36 and 79 steps, this might because we use the pre-

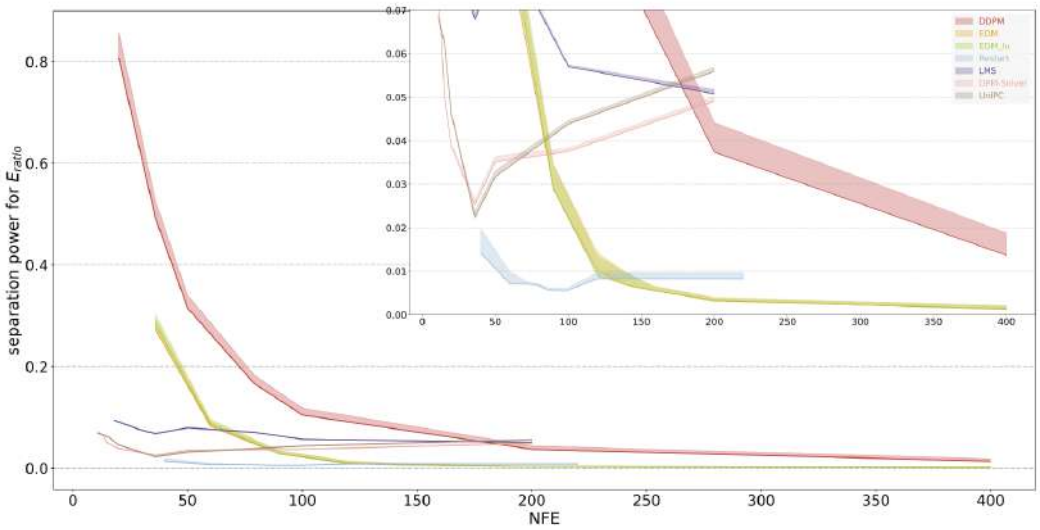

Figure 8: Separation power for $\boldsymbol{E_{\text{ratio}}}$ with different samplers and schedulers as a function of evaluation steps, the shaded area is possible values of separation power for 20 independent samplings.

defined configuration specifically for those steps. All proposed samplers have comparable and better performance with smaller steps. Most of the samplers we use actually have more parameters to tune than the baseline DDPM. For the common ODE samplers, a closer matching noise level during sampling would often yield better results. ~~, LMS sampler involves an additional parameter "order" of the coefficient which makes the generation time longer as it increases.~~ A possible future development is to fine-tune the ~~configuration~~ scheduler configuration for ODE samplers in higher steps.

~~AUC values for different samplers as a function of sampling steps, the shaded area is possible AUC value range for 20 evaluations through independent classifier training.~~

~~Separation power for $\boldsymbol{E_{\text{ratio}}}$ with different samplers and schedulers as a function of sampling steps, the shaded area is possible values of separation power for 20 independent samplings.~~

~~The distribution for total shower energy divided by the incident particle energy by left: 400 steps DDPM, middle: 30 steps Restart samplers, left: high step EDM samplers~~

Now we take a look at one of the most difficult and important quantities for the low level voxel diffusion – $\boldsymbol{E_{\text{ratio}}}$. In Fig.8, we choose 6 cases in our study. First, much faster convergences have been seen from ~~all~~ new introduced samplers by 5-20 times. The obtained result matches with the Table.3 mentioned in Song's [61]. For EDM ~~Heun SDE~~ sampler, the separation power, defined as the triangular discriminator [8] $\frac{\sum_{i=1}^{n}\left(\text{count}_{\text{data},i}-\text{count}_{\text{gen},i}\right)^2}{\sum_{i=1}^{n}\left(\text{count}_{\text{data},i}+\text{count}_{\text{gen},i}\right)}$ which calculated from each bins of the histogram, quickly drops around 60 steps and achieves better performance than the baseline 400 steps DDPM ~~.~~ sampler. The 20-30 steps DPM-Solver and UniPC samplers already have lower separation powers than 200 steps DDPM sampler. The 18 steps Restart sampler already could match the similar performance as the baseline. If we zoom-in the figure, at around 100-200 steps, ~~the separation power becomes~~ EDM and Restart samplers have 3-5 times weaker separation power than the baseline DDPM samplers. ~~In Fig. ??, we have seen really aligned shape between generated and reference classes for Restart and EDM samplers. As the steps goes even higher, the discrepancy between two histogram become smaller while the shape remains aligned. The histogram geneated from Restart samplers does not fit perfectly on lower ratio region, but still better than DDPM in other regions. This is crucial~~

Samplers(steps)	AUC ↓	Separation power (10^{-2}) ↓	FPD ↓
DDPM(79)	~~55.3~~ 61.45±0.05	~~0.0810~~ 8.10±1.05	0.074±0.01
DDPM(200)	~~53.2~~ 55.72±0.03	~~0.0344~~ 3.44±0.40	0.046±0.007
~~*DDPM(400)*~~ DDPM(400)	~~*52.6*~~ 54.6±0.02	~~*0.0155*~~ 1.55±0.30	~~*0.043*~~0.043±0.007
EDM(36)	~~54.1~~ 55.3±0.06	~~0.0256~~ 2.56±0.72	0.035±0.007
EDM(79)	**~~52.0~~52.9±0.04**	~~0.0076~~ 0.76±0.20	**0.027±0.004**
EDM(200)	~~51.5~~ 52.6±0.03†	~~0.0055~~ 0.55±0.10†	0.023±0.004‡
EDM(400)	~~51.1~~ 52.6±0.01*	**~~0.0039~~0.16±0.05***	0.021±0.003*
EDM_DPM++(79)	~~52.3~~ 53.0±0.04	~~0.0103~~ 0.82±0.20	0.026±0.004
EDM_Lu(79)	~~52.2~~ 52.8±0.04	~~0.0086~~ 0.86±0.20	0.026±0.004
Restart(18)	~~55.2~~ 56.7±0.07	~~0.0169~~ 1.69±0.66	0.059±0.009
Restart(36)	**~~52.0~~54.0±0.06**	~~0.0073~~ 0.73±0.26	**0.025±0.003**
Restart(79)	~~51.8~~ 53.9±0.05‡	~~0.0057~~ 0.57±0.19‡	0.022±0.004†
LMS(36)	~~53.8~~ 56.0±0.04	~~0.0305~~ 6.65±1.35	0.095±0.01
DPM++(~~20~~25)	~~59.8~~ 56.8±0.07	~~0.0534~~ 2.52±0.51	~~0.146~~0.113±0.01
UniPC(~~20~~25)	~~60.3~~ 56.9±0.07	~~0.1304~~ **2.31±0.45**	~~0.152~~0.112±0.01

Table 1: Summary table for high-level feature classifier mean AUC, E_{ratio} separation power, and Frechét Particle Distance [60] of of different samplers. ~~(~~Underline is the 400-step DDPM sampler (CaloDiffusion) as baseline~~)~~, the ⋆, †, ‡, are the first, second, third best results. **Bold** is the highlighted result with either great performance or good quality-sampling time balance.

~~for us to perform accurate energy calibration from low-level fast calorimeter simulation later.~~

The preliminary Frechét Particle Distance (FPD) [60] values for some samplers are also provided in Table. 1. The FPD metric is relatively biased and unstable among ~~sub samples~~ subsamples we choose. So we follow the way from [23] to subtract the non-zero FPD value between two `GEANT4` sample. Surprisingly, we find the EDM sampler converges even faster in this metric, surpassing the DDPM sampler around 36 steps ~~. The 18-step~~ (72 NFE). The 18-steps (∼ 40 NFE) Restart sampler has slightly worse performance, this is mainly because we choose the predefined configuration for smallest E_{ratio} separation power, a throughout fine-tune can make FPD value 20-30 percent smaller with almost same generation time. However, it still obtains the best FPD results with larger steps.

~~We choose Karras and Lu schedulers to illustrate the impacts of different noise schedulers on the same samplers. Changing the Karras scheduler to Lu will decrease the separation power~~ The separation power is decreased in smaller steps ($\lesssim$ **100**) but ~~increase it~~ increased in larger steps by changing the EDM scheduler from Karras to Lu. The choice of schedulers provides a training-free method to balance ~~and control~~ the quality and speed of the backward diffusion process. The EDM/DPM-Solver combined sampler with Karras scheduler performs pretty similarly as the EDM sampler with Lu scheduler. The best performances of Restart samplers again are achieved in 30-36 steps. ~~It~~ This sampler is crafted for efficiently conducting fast ~~ODE sampling~~ generation with outstanding performance and can be fine-tuned with post-training configuration. The DPM-solver exhibits a rapid drop in separation power within 10-20 steps but ceases to improve beyond those steps, achieving better performance than the 100-step DDPM around the 15$^{\text{th}}$ step. The plot further confirms that ODE sampler does better in low sampling steps as the discretization error of them are smaller than SDE ones in this region, the contraction of accumulated errors from SDE sampling process become much stronger than ODE so that a better convergence observed in higher sampling steps than ODE ones [49]. Some ODE samplers, such as LMS and DPM-solver, perform well on some metrics but not all.

There is not a clearly superior option that can achieve the best quality while having the lowest generation time.

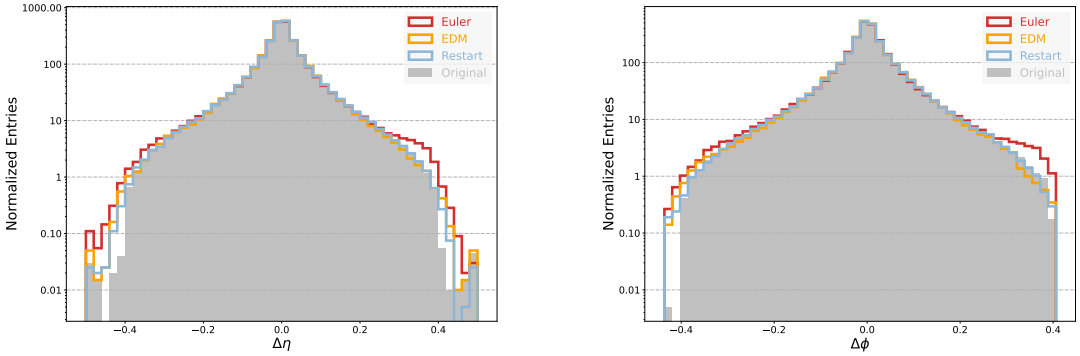

Figure 9: Comparison plots for relative angular coordinate of particle constituents inside gluon initiated jets with 200 steps DDIM Euler Sampler, 18 steps multi-level Restart Sampler, 50 steps EDM samplers . left: $\boldsymbol{\Delta\eta}$, right: $\boldsymbol{\Delta\phi}$.

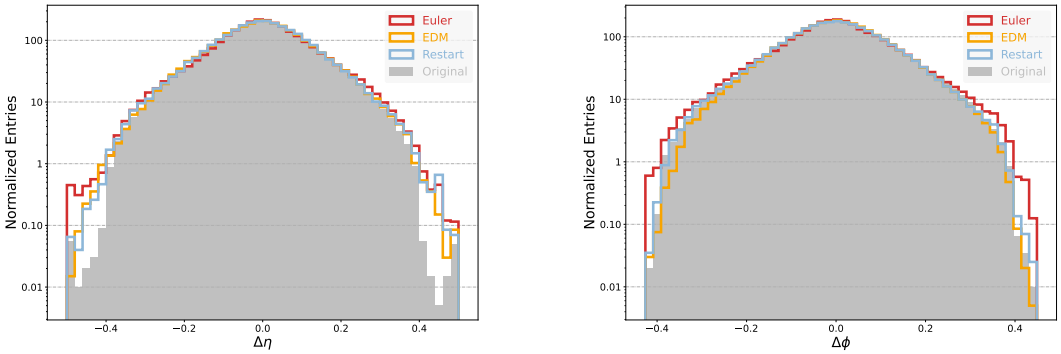

Figure 10: Comparison plots for relative angular coordinate of particle constituents inside top quark jets with 200 steps DDIM Euler Sampler, 18 steps multi-level Restart Sampler, 50 steps EDM samplers . left: $\boldsymbol{\Delta\eta}$, right: $\boldsymbol{\Delta\phi}$.

5.2 JetNet

We employ ~~the same~~ network training techniques with the same continuous noise level and representative samplers for the *JetNet* dataset to demonstrate that these methods are entirely portable across various datasets. As depicted in Fig. 9 ~~,~~ and 10, the low level feature generations with a ~~50-step~~ 50-steps EDM sampler and ~~18-step~~ 18-steps Restart sampler show better performance than a ~~200-step~~ 200-steps DDIM [36] Euler sampler on gluon jets. For more complicated topology, EDM and Restart have advantages in simulating the tails of the distribution over large η region. EDM sampler has small deficiency for underestimating the particles from top quark jets after $\boldsymbol{\Delta\phi > 0.2}$. This would have impact on reconstructing the jet level variables ~~as shown~~ in Fig. 11. ~~When~~

The same training setup in PC-JeDi [25] is used with DDPM samplers as baseline. We found the Euler samplers could give a better performance with our new proposed training. Both EDM and Restart samplers capture most of the high level gluon jet features, with only

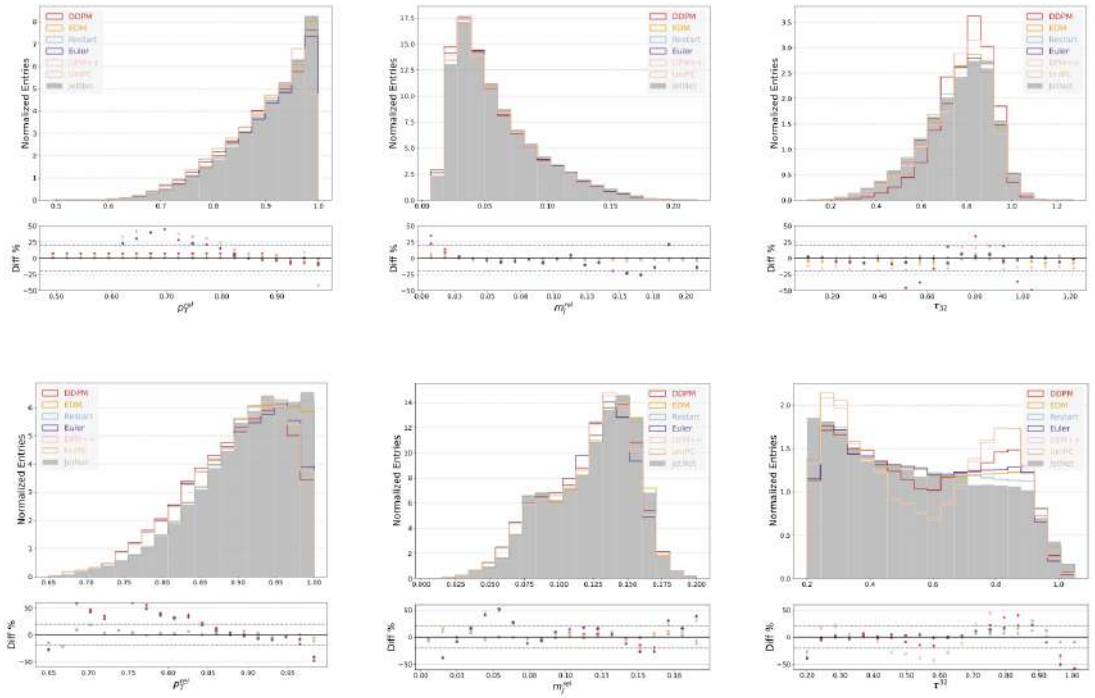

Figure 11: Comparison plots for ~~relative jet mass with 200 steps DDIM Euler Sampler, 18 steps multi-level Restart Sampler, 50 steps EDM samplers . left:~~ substructure of particle constituents inside gluon~~initiated jet, right:~~ /top quark ~~jet~~jets with different samplers and schedulers.

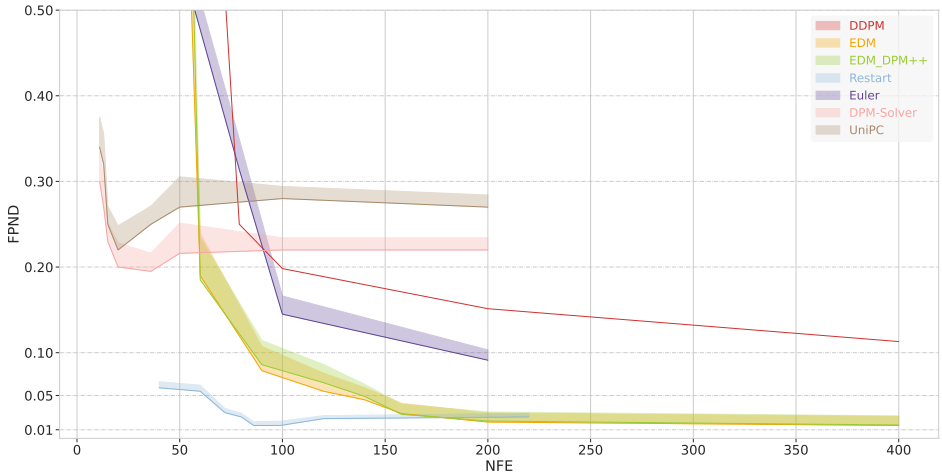

Figure 12: Comparison plots for FPD values for top jets as a function of evaluation steps with different samplers and schedulers. The shaded area is possible values of separation power for 10 independent samplings.

Samplers(~~steps~~NFE)	AUC ~~(Gluon Jet~~↓	FPD ↓	$W_m(10^{-4})$	$W_{p_T}(10^{-4})$	W
		Top			
DDPM(200)	~~FPD (Gluon Jet~~52.8±0.04	0.16±0.03	13.54	12.04	
Euler(200)	~~AUC (Top Jet~~52.6±0.04	0.15±0.03	20.47	10.08	
EDM(100)	~~FPD (Top Jet)~~ 51.9±0.03†	0.07±0.01†	7.65†	7.07†	
EDM_DPM++(100)	52.0±0.02‡	0.08±0.005‡	8.05‡	7.17‡	
Restart(70)	**51.3±0.03***	**0.02±0.005***	4.30*	5.65*	
DPM++(25)	53.7±0.05	0.20±0.04	25.64	10.45	
UniPC(25)	54.0±0.07	0.22±0.03	27.89	10.49	
		Gluon			
DDPM(200)	52.5±0.04	0.10±0.03	5.69	6.01	
Euler(200)	52.6±0.04	~~0.15~~ 0.14±0.04	~~53.5~~ 6.40	~~0.19~~ 6.63	
EDM(~~50~~100)	~~51.7 0.05~~ 51.9±0.03†	0.07±0.01†	5.03†	4.65†	
~~Restart~~EDM_DPM++(~~18~~100)	~~51.6~~ 52.0±0.03‡	~~0.04~~ 0.08±0.01‡	~~51.5~~ 5.40‡	4.59‡	
Restart(70)	**51.2±0.02***	**0.02±0.007***	**3.03***	3.20*	
DPM++(25)	52.8±0.05	**0.16±0.02**	6.05	9.97	
~~Restart_multi(18~~UniPC(25)	~~51.2~~ 52.9±0.05	~~**0.02**~~ 0.16±0.02	~~51.2~~ 6.15	~~**0.02**~~ 9.98	

Table 2: Summary table for low-level feature classifier mean AUC~~and~~, Frechét Particle Distance calculated from jet-level classifier, and high-level features unbinned (Wasserstein) distance of different samplers on gluon-initiated jets top and gluon quark jets. Underline is the 400-step DDPM sampler (PC-JeDi) as baseline, the ⋆, †, ‡, are the first, second, third best results. **Bold** is the highlighted result with either great performance or good quality-sampling time balance.

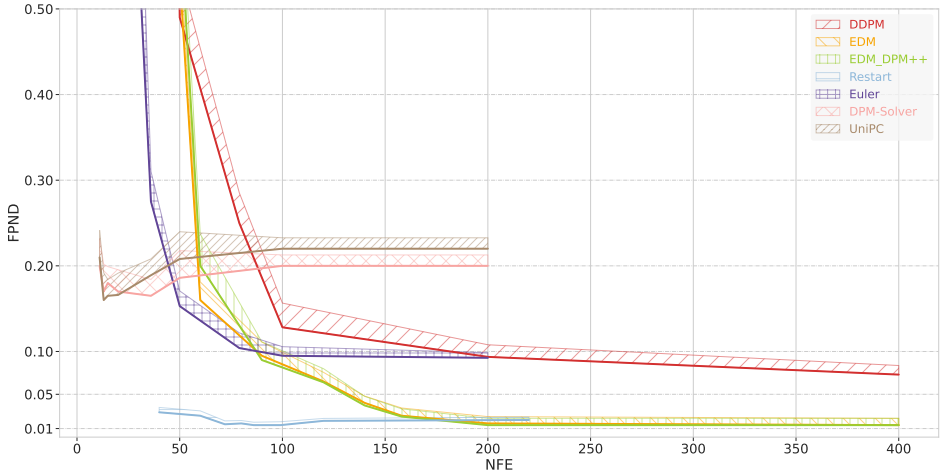

Figure 13: Comparison plots for FPD values for gluon jet as a function of evaluation steps with different samplers and schedulers. the shaded area is possible values of separation power for 10 independent samplings.

few percent difference in relative transverse momentum and mass of jets. The improvement is clearer in more challenging substructure variables such as the n-subjetiness between subleading and third leading jets - τ_{32} when comparing with default DDPM and Restart samplers. Although some ODE samplers outperform than DDPM on τ_{32} of gluon jets, they can not achieve better overall performance, except LMS samplers. The largest discrepancy for this quantity appears around 0.7-0.9 region where the baseline DDPM sampler has around 30-40% difference. The low-steps Restart sampler gives obviously better generated distribution than others with less than 10% difference over most regions.

While simulating the complex distribution in a highly boosted environment, ~~the EDM and DDIM Euler~~ most of the samplers fail to achieve satisfactory alignment around the two asymmetric peaks in the relative ~~jet mass distribution . The low step Restart sampler gives obviously better generated distribution than others. We evaluate the~~ mass distribution for top jets. We then anatomise the performance of proposed samplers via the three low level features with the same DNN classifier ~~, and~~ used in previous section, the values for the Fréchet Particle distance (FPD) calculated from the ~~Particle~~ ParticleNet [62] multi-classifier with N = ~~6.~~ 6, and the unbinned metrics (Wasserstein distance) for reconstructed individual high level features. Performances are summarized on Table. 2.

~~The~~ The Restart sampler provides the consistently best results for both top and gluon jets in all three metrics. Different scheduler variants on SDE samplers should not have too much influence on results. The 25-steps DPM-Solver sampler shows competitive result as the 200-steps traditional Euler and DDPM samplers on gluon jets.

In Fig. 12- 13, Restart sampler yields the best performance even in small number of function evaluations. Even though the boosts brought by those changes isn't as pronounced as they are in *CaloChallenge*~~. It may be because methods are more applicable to UNet and pixelated data than point clouds network~~, high step EDM and Restart samplers still have much lower FPD values than other samplers.

6 Conclusion and outlooks

In this study, we conducted a comprehensive investigation into the utilization of various state-of-the-art samplers and schedulers across diverse datasets in collider physics. We introduced a soft version of loss weighting techniques to expedite the convergence of physics-informed quantities towards a better reliable result. Our study presents several approaches for evaluating generated samples across different scenarios, providing a range of options for accelerating diffusion model without additional training or achieving superior results with fewer enhancements. Innovative sampling methods, such as Restart, exhibit better performance with just 18-30 steps, leading to a $\mathcal{O}(10)$ times speeding up without a significant performance loss. We showcase a potentially new baseline for training and sampling diffusion models in this rapidly evolving field.

Future development will aim to identify the optimal sets for these samplers on each dataset to achieve the maximum possible boost. Additionally, exploration into other areas, such as score distillation sampling and model fine-tuning, will be undertaken to obtain similar performance levels with even faster simulations.

Acknowledgements

We thank Kevin Pedro and Oz Amram for fruitful discussions about details of `CaloDiffusion` project, and insightful comments for different evaluation metrics.

References

[1] L. Evans and P. Bryant, *LHC Machine*, Journal of Instrumentation **3**(08), S08001 (2008), doi:10.1088/1748-0221/3/08/S08001.

[2] G. Aad *et al.*, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3**, S08003 (2008), doi:10.1088/1748-0221/3/08/S08003.

[3] S. Chatrchyan *et al.*, *The CMS Experiment at the CERN LHC*, JINST **3**, S08004 (2008), doi:10.1088/1748-0221/3/08/S08004.

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

[5] S. Amoroso *et al.*, *Challenges in Monte Carlo Event Generator Software for High-Luminosity LHC*, Comput. Softw. Big Sci. **5**(1), 12 (2021), doi:10.1007/s41781-021-00055-1, 2004.13687.

[6] G. Apollinari, O. Brüning, T. Nakamoto and L. Rossi, *High Luminosity Large Hadron Collider HL-LHC*, doi:10.5170/CERN-2015-005.1 (2015).

[7] M. Paganini, L. de Oliveira and B. Nachman, *CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks*, Physical Review D **97**(1) (2018), doi:10.1103/physrevd.97.014021.

[8] C. Krause and D. Shih, *Fast and accurate simulations of calorimeter showers with normalizing flows*, Physical Review D **107**(11) (2023), doi:10.1103/physrevd.107.113003.

[9] R. Zhang, *AtlFast3, the ATLAS fast simulation tool in Run3* (2023).

[10] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, W. Korcari, K. Krüger, P. McKeown and L. Rustige, *Hadrons, better, faster, stronger* (2021), 2112.09709.

[11] I. Pang, J. A. Raine and D. Shih, *Supercalo: Calorimeter shower super-resolution* (2024), 2308.11700.

[12] M. F. Giannelli and R. Zhang, *Caloshowergan, a generative adversarial networks model for fast calorimeter shower simulation* (2023), 2309.06515.

[13] S. Hoque, H. Jia, A. Abhishek, M. Fadaie, J. Q. Toledo-Marín, T. Vale, R. G. Melko, M. Swiatlowski and W. T. Fedorko, *Caloqvae : Simulating high-energy particle-calorimeter interactions using hybrid quantum-classical generative models* (2023), 2312.03179.

[14] J. Birk, E. Buhmann, C. Ewen, G. Kasieczka and D. Shih, *Flow matching beyond kinematics: Generating jets with particle-id and trajectory displacement information* (2023), 2312.00123.

[15] S. Bieringer, A. Butter, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, B. Nachman, T. Plehn and M. Trabs, *Calomplification — the power of generative calorimeter models*, Journal of Instrumentation **17**(09), P09028 (2022), doi:10.1088/1748-0221/17/09/p09028.

[16] M. A. W. Scham, D. Krücker, B. Käch and K. Borras, *Deeptreegan: Fast generation of high dimensional point clouds* (2023), 2311.12616.

[17] K. Dohi, *Variational autoencoders for jet simulation* (2020), 2009.04842.

[18] C. Krause and D. Shih, *Caloflow ii: Even faster and still accurate generation of calorimeter showers with normalizing flows* (2023), 2110.11377.

[19] C. Krause, B. Nachman, I. Pang, D. Shih and Y. Zhu, *Anomaly detection with flow-based fast calorimeter simulators* (2023), 2312.11618.

[20] A. Butter, T. Jezo, M. Klasen, M. Kuschick, S. Palacios Schweitzer and T. Plehn, *Kicking it Off(-shell) with Direct Diffusion* (2023), 2311.17175.

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

[22] V. Mikuni and B. Nachman, *Score-based generative models for calorimeter shower simulation*, Physical Review D **106**(9) (2022), doi:10.1103/physrevd.106.092009.

[23] O. Amram and K. Pedro, *Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation* (2023), 2308.03876.

[24] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown, *Caloclouds: Fast geometry-independent highly-granular calorimeter simulation* (2023), 2305.04847.

[25] M. Leigh, D. Sengupta, G. Quétant, J. A. Raine, K. Zoch and T. Golling, *Pc-jedi: Diffusion for particle cloud generation in high energy physics* (2023), 2303.05376.

[26] E. Buhmann, C. Ewen, D. A. Faroughy, T. Golling, G. Kasieczka, M. Leigh, G. Quétant, J. A. Raine, D. Sengupta and D. Shih, *Epic-ly fast particle cloud generation with flow-matching and diffusion* (2023), 2310.00049.

[27] V. Mikuni and B. Nachman, *Caloscore v2: Single-shot calorimeter shower simulation with diffusion models* (2023), 2308.03847.

[28] T. Salimans and J. Ho, *Progressive distillation for fast sampling of diffusion models* (2022), 2202.00512.

[29] E. Buhmann, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown, *Caloclouds ii: Ultra-fast geometry-independent highly-granular calorimeter simulation* (2023), 2309.05704.

[30] Y. Song, P. Dhariwal, M. Chen and I. Sutskever, *Consistency models* (2023), 2303.01469.

[31] M. Leigh, D. Sengupta, J. A. Raine, G. Quétant and T. Golling, *Pc-droid: Faster diffusion and improved quality for particle cloud generation* (2023), 2307.06836.

[32] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, *High-resolution image synthesis with latent diffusion models* (2022), 2112.10752.

[33] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein and K. Aberman, *Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation* (2023), 2208.12242.

[34] L. Zhang, A. Rao and M. Agrawala, *Adding conditional control to text-to-image diffusion models* (2023), 2302.05543.

[35] S. Zhai, Y. Dong, Q. Shen, S. Pu, Y. Fang and H. Su, *Text-to-image diffusion models can be easily backdoored through multimodal data poisoning* (2023), 2305.04175.

[36] J. Song, C. Meng and S. Ermon, *Denoising diffusion implicit models* (2022), 2010.02502.

[37] M. F. Giannelli, G. Kasieczka, B. N. Claudius Krause, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022* (2022).

[38] C. Krause, M. F. Giannelli, G. Kasieczka, B. Nachman and et al., *Calochallenge 2022: A community challenge for fast calorimeter simulation* (2024), 2410.21611.

[39] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant and D. Gunopulos, *Particle cloud generation with message passing generative adversarial networks* (2022), 2106.11535.

[40] S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman and D. Shih, *Dctrgan: improving the precision of generative models with reweighting*, Journal of Instrumentation **15**(11), P11004–P11004 (2020), doi:10.1088/1748-0221/15/11/p11004.

[41] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics* (2015), 1503.03585.

[42] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui and M.-H. Yang, *Diffusion models: A comprehensive survey of methods and applications* (2023), 2209.00796.

[43] B. B. Moser, A. S. Shanbhag, F. Raue, S. Frolov, S. Palacio and A. Dengel, *Diffusion models, image super-resolution and everything: A survey* (2024), 2401.00736.

[44] J. Ho, A. Jain and P. Abbeel, *Denoising diffusion probabilistic models* (2020), 2006.11239.

[45] A. Nichol and P. Dhariwal, *Improved denoising diffusion probabilistic models* (2021), 2102.09672.

[46] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon and B. Poole, *Score-based generative modeling through stochastic differential equations* (2021), 2011.13456.

[47] T. Karras, M. Aittala, T. Aila and S. Laine, *Elucidating the design space of diffusion-based generative models* (2022), 2206.00364.

[48] ~~C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li and J. Zhu, Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps(2022), .~~

[49] Y. Xu, M. Deng, X. Cheng, Y. Tian, Z. Liu and T. Jaakkola, *Restart sampling for improving generative processes* (2023), 2306.14878.

[50] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li and J. Zhu, *Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models* (2023), 2211.01095.

[51]

[52] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li and J. Zhu, *Dpm-solver~~++~~: ~~Fast~~ A fast ode solver for ~~guided sampling of~~ diffusion probabilistic ~~models~~model sampling in around 10 steps* (~~2023),~~ 2022), 2206.00927.

[53] *Runge–Kutta Methods*, chap. 3, pp. 137–316, John Wiley & Sons, Ltd, ISBN 9780470753767, doi:https://doi.org/10.1002/9780470753767.ch3 (2008).

[54] W. Zhao, L. Bai, Y. Rao, J. Zhou and J. Lu, *Unipc: A unified predictor-corrector framework for fast sampling of diffusion models* (2023), 2302.04867.

[55] M. Cacciari, G. P. Salam and G. Soyez, *The anti-ktjet clustering algorithm*, Journal of High Energy Physics **2008**(04), 063–063 (2008), doi:10.1088/1126-6708/2008/04/063.

[56] P. Dhariwal and A. Nichol, *Diffusion models beat gans on image synthesis* (2021), 2105.05233.

[57] C. Meng, R. Rombach, R. Gao, D. P. Kingma, S. Ermon, J. Ho and T. Salimans, *On distillation of guided diffusion models* (2023), 2210.03142.

[58] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi and D. J. Fleet, *Video diffusion models* (2022), 2204.03458.

[59] T. Hang, S. Gu, C. Li, J. Bao, D. Chen, H. Hu, X. Geng and B. Guo, *Efficient diffusion training via min-snr weighting strategy* (2023), 2303.09556.

[60] R. Kansal, A. Li, J. Duarte, N. Chernyavskaya, M. Pierini, B. Orzari and T. Tomei, *Evaluating generative models in high energy physics*, Physical Review D **107**(7) (2023), doi:10.1103/physrevd.107.076017.

[61] Y. Song and P. Dhariwal, *Improved techniques for training consistency models* (2023), 2310.14189.

[62] H. Qu and L. Gouskos, *Jet tagging via particle clouds*, Physical Review D **101**(5) (2020), doi:10.1103/physrevd.101.056019.

A Hyperparameters of Samplers and Schedulers

We will list more details about the hyperparamter range we ended up choosing for each sampler.

Dataset	S_{Churn}	S_{max}	S_{min}	S_{noise}	σ_{max}	σ_{min}
CaloCha.	[15-40]	[20-40]	[0.01-0.1]	[1.001-1.004]	[40-80]	[0.002-0.01]
JetNet	[20-50]	[20-39]	[0.01-0.1]	[1.001-1.006]	[30-80]	[0.002-0.01]

Table 3: Summary table for Stochastic parameter and scheduler of EDM Heun Karras sampler

For the Stochastic parameter we listed on Table. 3, the choices of S_{Churn} and S_{min} are very important. In the *CaloChallenge* dataset, a too small S_{Churn} would make the prediction worse over lower energy region, resulting a higher E_{ratio} separation power, and a high S_{Churn} would make noise injection power too high over low sampling step region to converge quickly. With our training setup, we usually set a larger S_{Churn} on the *Jetnet* dataset. That is also one reason why a smaller improvement is observed here. The S_{noise} does not affect too much on both dataset.

We also find several ways to constrain the multilevel configuration for generating satisfied results with Restart sampler by empirical evaluations.

Oftentimes different N_{steps} across different level (range of time steps) yield better FPD value. Gradually increasing N_{steps} toward last few steps give slightly better result than constant N_{steps} among all levels. With a larger K iteration value in 3^{rd} and 5^{th}, we obtain the best results in current sampling steps, but also increase the generation time by few percent.

level	N_{steps}	K	σ_{max}	σ_{min}
0	[2-5]	[2-15]	[1.09,0.30]	[0.14, 0.06]
1	[2-5]	[1-2]	[40.79]	[19.35]
2	[3-6]	[1-2]	[1.92]	[1.09]
3	[4-6]	[2-4]	[1.09,0.59,0.30]	[0.59,0.06]
4	[4-6]	[1-4]	[0.59,0.30]	[0.30,0.06]
5	[4-7]	[2-6]	[0.30]	[0.06]

Table 4: Summary table for good parameter of Restart Karras sampler, 0 denoted as the single level Restart.

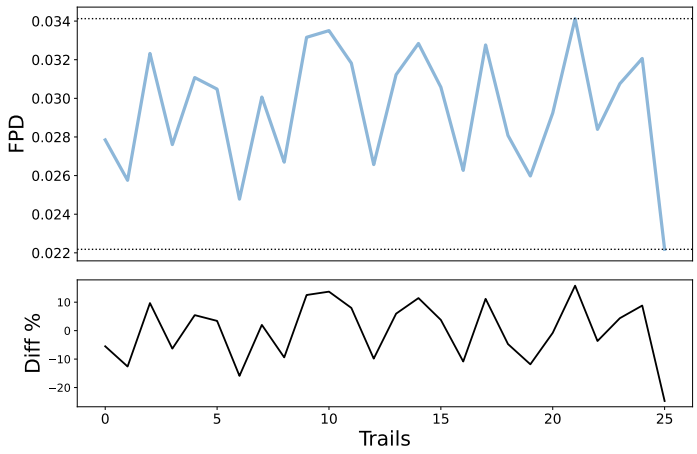

Figure 14: FPD values in CaloChallenge for 36 steps multilevel Restart samplers with 25 different configurations

For DPM-solver, we choose a slightly different noise level. The $\boldsymbol{\beta_{\max}}, \boldsymbol{\beta_{\min}}$ for VP scheduler is $\mathbf{19.9, 0.002}$. For Karras and Lu scheduler, $\boldsymbol{\sigma_{\max}}, \boldsymbol{\sigma_{\min}}$ is $\mathbf{[15, 40], [0.001, 0.01]}$. For LMS, we choose $\mathbf{[3, 4, 6]^{th}}$ order of calculation. The UniPC sampler is using $\mathbf{2^{nd}}$ order $\boldsymbol{B(h)}$-$\mathbf{2}$ solver with the same scheduler as DPM-solver.