DId not mention about two step method. For Evaluation, used average shower energy, distribution of center, width and layer energy, voxel energy, dep/gen energy,mean dep energy., Classifier test

FERMILAB-PUB-23-384-CSAID-PPD

Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation

Oz Amram[1,*] and Kevin Pedro[1,†]

[1] *Fermi National Accelerator Laboratory, Batavia, IL 60510, USA*

Simulation is crucial for all aspects of collider data analysis, but the available computing budget in the High Luminosity LHC era will be severely constrained. Generative machine learning models may act as surrogates to replace physics-based full simulation of particle detectors, and diffusion models have recently emerged as the state of the art for other generative tasks. We introduce `CaloDiffusion`, a denoising diffusion model trained on the public *CaloChallenge* datasets to generate calorimeter showers. Our algorithm employs 3D cylindrical convolutions, which take advantage of symmetries of the underlying data representation. To handle irregular detector geometries, we augment the diffusion model with a new geometry latent mapping (`GLaM`) layer to learn forward and reverse transformations to a regular geometry that is suitable for cylindrical convolutions. The showers generated by our approach are nearly indistinguishable from the full simulation, as measured by several different metrics.

The quantities shown are the width of the shower in the angular direction (left), the distribution of total number of non-zero voxels (center), and the energy per voxel (right).

I. INTRODUCTION

High quality simulation plays a crucial role in modern particle physics experiments. Most experiments rely on the `Geant4` [1–3] toolkit to simulate interactions of particles with their detector. Achieving accurate results requires simulating the interactions of both the primary particle incident on the detector and the numerous secondary particles produced through interactions with the detector material. For this reason, simulations of calorimeters, which are designed to capture the energy produced by the shower of secondary particles, usually requires the most computational resources. Simulating calorimeters currently consumes a significant fraction of the computing resources of modern collider experiments [4]. The problem will be exacerbated at the High Luminosity LHC, which will feature larger data volumes, more complex detectors [5], and a higher pileup environment. Future high granularity detectors will require more computational resources to simulate because of their more complex geometries and higher levels of precision [6]. At the same time, reconstruction will require a larger fraction of the computing budget because of the expected superlinear scaling of important algorithms with increasing pileup [7].

These resource constraints mean that full, detailed detector simulation using `Geant4` will not be possible for every simulated event. Instead, 'fast simulation' methods that approximate the output of `Geant4` using less computation will be employed. Most major experiments have developed fast simulation frameworks based on parametric approximations manually tuned to `Geant4` [8–13]. These parametric models generally suffer from deficiencies in modeling detailed observables of calorimeter showers, limiting their usage in physics analysis.

In order to overcome these challenges, machine learning (ML) models are increasing in popularity as fast surrogate models for `Geant4` [13–32] (see Ref. [33] for an overview and Ref. [34] for a recent review). These techniques borrow from the growing field of ML-based generative modeling, which has made significant advances in recent years.

In high energy physics (HEP), the first class of generative models proposed for this purpose were generative adversarial networks (GANs) [14]. GANs are trained by iterating between a 'generator network' that learns to produce artificial samples and a 'discriminator network' which attempts to distinguish the artificial samples from true ones. GANs are able to generate high quality showers orders of magnitude faster than `Geant4`. The ATLAS experiment has now employed calorimeter GANs in their fast simulation framework [13]. GANs are also used for fast simulation by the LHCb [35] experiment and are being explored for emulating the high granularity, 7.5M channel, pixel detector of Belle-II [27]. However, GAN training does not reliably converge because the two competing objectives create a saddle point in the loss space rather than a minimum. Additionally, GANs are known to suffer from 'mode collapse', in which the generator network only learns to produce samples from a subset of the full data space.

Variational Autoencoders (VAEs) have also been proposed for calorimeter simulation [17, 21, 22]. A VAE consists of an encoder, which maps the input data to a smaller latent space, and a decoder, which maps the latent space back to the original data. A VAE is distinguished from a regular autoencoder by forcing the latent space to follow a multivariate Gaussian distribution via an additional term in the training loss. New samples can then be generated by drawing random samples from a multivariate Gaussian in the latent space and applying the decoder model. However, VAEs on their own do not seem have the expressive power of GANs and other state-of-the-art models and generally achieve worse quality on complex high-dimensional data such as calorimeter

* oamram@fnal.gov
† pedrok@fnal.gov

showers. Refs. [17, 21] instead use a bounded information bottleneck AE (BIB-AE), which is a novel combination of the VAE and GAN architecture.

Normalizing flows (NFs) have also been proposed for calorimeter simulations [19, 26, 31]. NFs are based on a series of invertible transformations that convert the input distributions to multivariate Gaussians. Once trained, new samples can be generated by sampling the Gaussian space and applying the inverse transformations to convert to the data space. However, as the dimensionality of the data has to be preserved in each stage of the flow, it can be difficult to scale NFs to very high-dimensional data.

Recently, a new class of models has become dominant in ML image generation tasks: denoising diffusion models [36–38]. In this work, we explore the use of denoising diffusion models to generate calorimeter showers. Diffusion models are based on a 'noising process' that continuously perturbs an image until it is degraded to pure noise. A 'denoising model' is then trained to invert the diffusion process. New samples can be generated by constructing a sample in the noise space and repeatedly 'denoising' it back to the original space. The use of diffusion models in image generation has proliferated because of their straightforward training procedure, high quality results, straightforward scaling to high-dimensional data, and manageable computational requirements. Diffusion models were first used for calorimeter simulation in `CaloScore` [24, 32] with promising results. `CaloScore` is a score-based diffusion model, which is similar but distinct from the denoising diffusion model employed in this paper. Recent work has combined diffusion with point clouds [30, 39] and demonstrated distillation of diffusion models to improve generation time of jet particle clouds [29, 40]. Several other works apply diffusion to HEP in other contexts [41–43].

Our approach, dubbed `CaloDiffusion`, is a denoising diffusion model for calorimeter simulation and employs several novel optimizations to make use of the geometric structure of the data. In contrast to other recent works [30, 39], which have advocated for point cloud representations of calorimeter showers, `CaloDiffusion` uses voxelized image-like representations of the calorimeter data. The use of this voxelized representation retains the geometric information of the data, allowing for several optimizations that exploit the cylindrical structure and scale well for high-dimensional datasets. We additionally introduce a new geometry latent mapping (`GLaM`) component, which is able to map irregular detector geometries into a regular structure suitable for symmetry-preserving operations such as convolutions.

We test our approach on the public datasets provided as part of the Fast Calorimeter Simulation Challenge (*CaloChallenge*) [44]. The challenge released three datasets of showers simulated with `Geant4` in calorimeters with increasing granularity. We find that `CaloDiffusion` is able to generate very quality showers that are difficult to distinguish from `Geant4` for all datasets of the *CaloChallenge*. Based on quantitative metrics, we demonstrate significant gains over previous state-of-the-art methods, particularly for the high-dimensional datasets of the *CaloChallenge*.

II. DIFFUSION MODELS

Diffusion models are defined in terms of a 'noising process', which is a Markov chain that starts from data points x_0 (following a probability distribution $q(x_0)$) and iteratively adds Gaussian noise. The data points x_t at time t are generated from data points at the previous time step x_{t-1} by adding Gaussian noise ϵ. At the final time step T, the probability distribution of data points $q(x_T|x_0)$ can then be computed based on the original x_0 via a product of Gaussian likelihoods. This is summarized in the following equations:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \beta_t\epsilon, \tag{1}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t), \tag{2}$$

$$q(x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \tag{3}$$

where we denote Gaussian likelihoods as $\mathcal{N}(x|\mu, \sigma^2)$, $\epsilon \sim N(0, \mathcal{I})$, and β_t is a 'variance schedule' that controls how much Gaussian noise is added at each time step.

For a sufficiently large T (the total number of diffusion steps), the Gaussian noise will overwhelm the original data and x_T will follow a multivariate Gaussian distribution. Therefore, a new sample x_0 could be generated by sampling x_T from a multivariate Gaussian and inverting the diffusion process in order to produce $x_0 \sim q(x_0)$. An exact inversion of the diffusion process requires knowing the reverse distribution $p(x_{t-1}|x_t)$, which encodes how likely a particular data point x_{t-1} is given the noisier version x_t. Direct calculation of $p(x_{t-1}|x_t)$ could be done via Bayes' rule $p(x_{t-1}|x_t) = {}^{q(x_t|x_{t-1})q(x_{t-1})}/_{q(x_t)}$, but this is intractable because evaluating $q(x_t) = \int dx_0 q(x_0) \prod_{t=1}^{T} q(x_t|x_{t-1})$ requires an integral over the entire data distribution $q(x_0)$. We therefore approximate $p(x_{t-1}|x_t)$ as:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}|\mu_\theta(x_t, t, z), \beta_t\mathcal{I}), \tag{4}$$

where the estimated mean μ_θ is modeled by a neural network with parameters θ, conditioned on t and additional information z. There are multiple ways to parameterize $\mu_\theta(x_t, t, z)$ and we employ two different approaches as discussed in Section IV B.

Because sums of Gaussians also follow a Gaussian distribution, x_t can be directly sampled from x_0 in a single step:

$$q(x_T|x_0) = \mathcal{N}(x_t|\sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)\mathcal{I}) \tag{5}$$

$$x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon \tag{6}$$

where $\alpha_t \equiv 1-\beta_t$ and $\overline{\alpha}_t = \prod_{\tau=1}^{t} \alpha_\tau$. The variance of the noise for time step t is therefore $1-\overline{\alpha}_t$, which can be used to define the noise schedule as an alternative to β_t. This property is convenient because efficiently computing x_t from x_0 allows t to be randomly sampled during training.

Training a denoising diffusion model proceeds via the following steps: sampling a batch of images x' from the training set; choosing random time steps t'; producing a set of noised images $x'_{t'}$ based on Eq. (6); and comparing the model's prediction for μ_θ to the true value to compute the loss. Once the model has been trained, new samples can be generated by first sampling $x_T \sim \mathcal{N}(0, \mathcal{I})$, then repeatedly evaluating $p(x_{t-1}|x_t)$ from the trained model until x_0 is reached.

The denoising diffusion approach employed here shares many features with score-based diffusion, or score-matching models, such as `CaloScore` [24, 32]. The score-based approach defines a stochastic differential equation (SDE) that continuously corrupts the data into a known distribution. Rather than directly learning to invert the denoising process, the neural network is trained to evaluate the score of the data, $\nabla_x \log q(x)$, which can then be used to reverse the SDE in order to generate new samples. There are different ways to parameterize the SDE, based on the choices of the 'diffusion' and 'drift' functions. However, the 'variance preserving' formulation is deeply tied to the denoising diffusion approach employed here: the optimal score-matching network is identical to the optimal denoising network (see Appendix B of Ref. [38] for a short derivation). Both score-based and denoising-based diffusion models are being actively explored in the ML literature [38]. We focus on the denoising variant here because of its conceptual simplicity.

III. DATASETS

To facilitate a comparison with other work, we test our methods on the datasets of the *CaloChallenge*. The first dataset from the *CaloChallenge* consists of voxelized showers from single particles, γ or $\pi^\pm$, interacting with the ATLAS detector in the η range [0.2, 0.25] [45]. 15 different incident particle energies, spanning the range 256 MeV up to 4 TeV in powers of 2, are included. 10,000 events per incident energy are provided, except for the highest energies, which have fewer events and therefore higher statistical uncertainty. In total, 242000 (241600) events are provided for the photon (pion) dataset. These datasets were used by ATLAS to train the `FastCaloGAN` [16] model used in `AltFast3` [13]. The voxelized representations have 5 and 7 layers with 368 and 533 voxels, respectively, for the γ and $\pi^\pm$ showers. There are different numbers of angular and radial bins within each layer to reflect the varying granularity of the ATLAS calorimeter. For the photon (pion) datasets, layers 1 and 2 (1, 2, 12, and 13) have 10 angular bins and the rest have only a single angular bin. Each layer has a unique binning in the radial direction. For example, the first layer of the pion dataset has 8 variable-width bins covering a radial distance up to 600 cm, while the last layer has 10 variable-width bins covering up to 2000 cm. Because of the unique binning in each layer, only two bins from the first layer exactly align with a bin from the last layer. 'There are a total of 30 (23) unique radial bin edges for the photon (pion) dataset.

Datasets 2 and 3 of the *CaloChallenge* each consist of 200,000 showers from an electron incident on a cylindrical sampling calorimeter with 45 layers, each with an active (silicon) and passive (tungsten) component. The electron energy spans the range of 1 GeV to 1 TeV with a log-uniform distribution. Each layer in dataset 2 has 9 radial bins and 16 angular bins, leading to a total of $45 \times 16 \times 9 = 6480$ voxels in each shower. Dataset 3 features a much higher granularity; each layer has 18 radial and 50 angular bins, leading to a total of $45 \times 50 \times 18 = 40500$ voxels in each shower.

Following the specifications of the *CaloChallenge*, we split the available events evenly between training and evaluation for all datasets. The resulting size of the training sample, only $O(100\text{K})$ showers, is relatively limited, especially for very high-dimensional data such as dataset 3. It is likely generating additional showers for training would lead to improved performance. However, if this limited sample is taken to represent only a small portion of a real particle detector geometry, it may be a realistic estimate of the practically achievable training sample size, given restrictions on available computing resources. For example, the approach employed by ATLAS for `FastCaloGAN` involves training a separate model for each of 100 different η regions of the detector and thus can only generate a limited number of events for each η region.

IV. METHODS

A. Preprocessing

We apply several stages of preprocessing to the showers before the diffusion process. First, the energy in each voxel is divided by the incident particle energy, yielding the normalized energy E_i in voxel i. As in previous work [19, 24], a 'logit' transformation is then applied to the voxel energies:

$$u_i = \log\left(\frac{x}{1-x}\right), \quad x = \delta + (1 - 2 * \delta) * E_i, \tag{7}$$

where $\delta = 10^{-6}$ avoids discontinuities at $x = 0$ and $x = 1$. We then subtract the mean and divide by the standard deviation of the transformed voxel energy distribution u_i:

$$u'_i = \frac{u_i - \overline{u}}{\sigma_u}. \tag{8}$$

The distribution of preprocessed voxel energies u' has zero mean and unit variance, which is important to en-

sure the signal-to-noise ratio during the diffusion process has the appropriate magnitude.

The incident particle energy is used as a conditioning input to the model. We first apply a logarithm to the energy and then scale the resulting values to fall in the range 0 to 1.

B. Diffusion Specifics

We train our model based on a diffusion process with 400 noising steps. We follow Ref. [46] and use a 'cosine' noise schedule, defined as:

$$\overline{\alpha}_t = \cos\left(\frac{\frac{t}{T}+s}{1+s}\cdot\frac{\pi}{2}\right) \tag{9}$$

with $s = 0.008$. This noise schedule adds noise more slowly during the intermediate steps of the diffusion process than the simple linear schedule originally used in Ref. [36]. This preserves information for longer during the process, and we find it reduces the number of diffusion steps needed to maintain high quality.

As mentioned in Section II, there are different choices for the parameterization of the training objective of the model. The most obvious approach is to predict the denoised image x_0 directly. Ref. [36] suggests predicting the normalized noise component, ϵ, and then computing μ_θ as:

$$\mu_\theta(x_t,t,z) = \frac{1}{\sqrt{\overline{\alpha}_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t,t)\right). \tag{10}$$

In this case, training proceeds by minimizing the loss:

$$\mathcal{L} = \mathbb{E}_{t,\epsilon}\left[||\epsilon_\theta(x_t,t,z)-\epsilon||^2\right]. \tag{11}$$

The argument in favor of predicting the normalized noise component as the training objective is that it allows the model output to stay in a consistent range, so the model learns to make subtle refinements when the noise levels are small. However, when the noise levels are large, small inaccuracies in the model prediction can lead to large changes to the image in the sampling process. This can be somewhat mitigated by skipping the first steps of the diffusion process during generation in order to avoid this divergent behavior. We find this parameterization works well for datasets 1 and 2.

For dataset 3, we find the training objective suggested by Ref. [38], where the model predicts a weighted average of the noise component and the denoised image, yields better results:

$$\mathcal{L} = \mathbb{E}_{t,\epsilon}\left[w(t)\left|\left|\; F_\theta(x_t,t,z) - \frac{1}{c_{\text{out}}(t)}\left(x_0 - c_{\text{skip}}(t)\cdot(x_t)\right)\right|\right|^2\right]. \tag{12}$$

The different weighting functions are chosen to be proportional to the standard deviation of the total amount of noise at each step t, $\sigma(t) = \sqrt{1-\overline{\alpha}_t}$. Specifically, $w(t) = 1 + 1/\sigma(t)^2$, $c_{\text{skip}}(t) = 1/(\sigma(t)^2+1)$, and $c_{\text{out}}(t) = 1/(1+1/\sigma(t)^2)$. With this combination of terms, the model trades off between predicting the noise component when the noise is small, and predicting the denoised image when the noise is large. For $t \to 0$, $\sigma(t) \to 0$ and $c_{\text{skip}} \to 1$, so the training objective of the model is roughly proportional to ϵ. But for $t \to T$, $\sigma(t) \to 1$ and $c_{\text{skip}} \to 1/2$, and the training objective is a weighted average of the denoised image x_0 and the noise component ϵ. This scheme makes the model less sensitive to inaccurate predictions at high noise levels during the sampling process. This effect is more important for dataset 3 because of its higher sparsity, which leads to a longer tail in the voxel energy distribution. When using this training objective, skipping the first iterations of the diffusion process when sampling is no longer required.

We follow the stochastic sampling algorithm proposed in Ref. [36], in which a small amount of additional noise is added back to the sample after each denoising step:

$$x_{t-1} = \frac{1}{\sqrt{\overline{\alpha}_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t,t)\right) + \sigma_t\epsilon' \tag{13}$$

for $\epsilon' \sim \mathcal{N}(0,\mathbb{I})$ and $\sigma_t = \beta_t(1-\overline{\alpha}_{t-1}/1-\overline{\alpha}_t)$.

As long as the model is conditioned on the noise level, the number of diffusion steps in the sampling need not be the same as the number of steps in the training. Decreasing the number of diffusion steps will linearly improve the computational time needed to generate samples, but may produce samples of lower quality. This provides significant flexibility in trading between sample quality and computation time for a trained model. As the optimal balance between sample quality and computation time will be application-specific, in this work we primarily focus on sample quality. For datasets 1-photon, 2 and 3 we choose 200 diffusion steps for sampling because we find it does not significantly degrade sample quality compared to 400 steps, but further reductions do. For datasetpions we find that 200 diffusion steps noticeably reduces the sample quality and therefore report results using 400 steps. Additionally, for datasets 1 and 2, we find that skipping the first two denoising steps (i.e. starting from x_{T-2} rather than x_T) avoids instabilities caused by imperfect estimates of ϵ at the highest noise levels.

After generation, we apply a cutoff on the minimum voxel energy to match the minimum value in the *CaloChallenge* datasets. This corresponds to a value of 10 MeV for dataset 1, and 15 keV for datasets 2 and 3. Voxels below this value are set to zero. We note that the threshold for datasets 2 and 3 is likely unrealistically low for a real detector operating in the energy range considered, however we use these values to maintain consistency with the *CaloChallenge*.

C. Network Architecture

The primary input to the network is the noisy representation of the shower. However, additional, conditional information is provided as well. The conditional information consists of the scaled logarithm of the incident energy of the particle and the noise level of the current diffusion step ($\sqrt{1-\overline{\alpha_t}}$). This conditional information is encoded into a 128 dimensional vector via a two-layer fully connected network.

The denoising model uses a U-net [47] architecture, which is commonly employed in diffusion tasks. U-net architectures resemble an encoder-decoder pattern, where the input is gradually compressed to a smaller space, but unlike an autoencoder, skip connections are used so that there is no information bottleneck. Our U-net has an initial convolution followed by a series of ResNet blocks [48]. Conditional convolutions are created by adding the conditional information as an additional bias term after the first convolutional layer of each ResNet block. For datasets 1 and 2 (3) we use three ResNet blocks for the encoder with 16/16/32 (32/32/32) filters. Convolutional layers with a stride of 2 and appropriate padding are used to reduce the data size by a factor of two in each dimension after each of the first two ResNet blocks. Linear self-attention layers [49] are applied after each ResNet block. The architecture is then mirrored, with three more ResNet blocks with the same filter sizes. Convolutional transpose layers are used to upsample by a factor of two after each ResNet block to return to the original data dimension. A schematic of the network architecture is shown in Fig. 1.

In total, the models for datasets 1 and 2 (dataset 3) consists of ∼520K (∼1.2M) parameters. The model architectures were not extensively optimized, and it is likely the performance could be further improved with a dedicated optimization procedure.

V. GEOMETRIC INNOVATIONS

A. Optimizations for Cylindrical Geometry

Regular convolutions achieve their power by exploiting the underlying symmetry of the data: translation invariance along each of the coordinate dimensions. When a convolutional layer is applied to an image, the filters perform the same local operation across the whole input image. This allows for expressive, parameter-efficient operations on high-dimensional images. However, while calorimeter showers represented in a voxelized cylindrical geometry have a regular structure, they are not inherently translation invariant. The distribution of energy deposited in each layer encodes important information about the shower, which would be spoiled by translating the shower in either direction along the layer axis. Likewise, the distribution of energy in the radial direction encodes important information about the transverse spread of the shower and falls rapidly as a function of the distance away from the shower center. Additionally, in a realistic detector, sensors in different layers may have different sizes or be made of different materials. The one coordinate dimension that may be translation invariant is the angular dimension. However, this dimension has a periodic topology that will not be respected by regular convolutions. We therefore design several novel optimizations of the convolution operation tailored to cylindrical data that improve the output fidelity.

In order to respect the periodicity of the angular dimension in cylindrical calorimeters, our denoising network uses cylindrical convolutions rather than standard Cartesian ones. The angular dimension is represented in a linear array, so neighboring values with coordinates near the extrema of the angular range are far apart in the array representation. Before each cylindrical convolution operation, a circular padding is added in the angular dimension, such that both ends of the linear array are extended with the values from the opposite end. This ensures that when a 3D convolution is applied, the voxels close to the ends of the linear array properly interact with their angular neighbors on the opposite end. This padding is only applied to the angular dimension; the radial and z dimensions remain unchanged.

To allow our convolutional operations to violate translation invariance, we devise a novel scheme for location-conditional convolutions. This is implemented by augmenting the shower image with additional input channels that encode the position of each voxel. We construct one 'layer image' in which the value of each voxel corresponds to the layer number of that voxel, normalized to the range 0 to 1. We similarly construct a 'radial image' that encodes the radial distance of each voxel, also normalized from 0 to 1. For dataset 1, we observe slight non-uniformities in the energy distribution as a function of the angular bin, and find slight performance gains from including an 'angular image' as well. These additional images are concatenated to the per-voxel shower energy as additional input channels. This allows the filters in the convolutional operations to produce different results in different parts of the geometry. The output of the denoising network is still a single channel corresponding to the energy in each voxel. As these images are the same for every input, in principle they do not supply any additional information to the model. Therefore, one would expect that they would be unnecessary for a sufficiently large and expressive model. However, in practice, with the models employed in this work, we have found this technique makes it easier for the network to learn the non-uniformities of the underlying data.

B. GLaM: Geometry Latent Mapping

Though datasets 2 and 3 feature significantly larger numbers of voxels than dataset 1, their regular binning allows convolutional operations to be readily applied. In

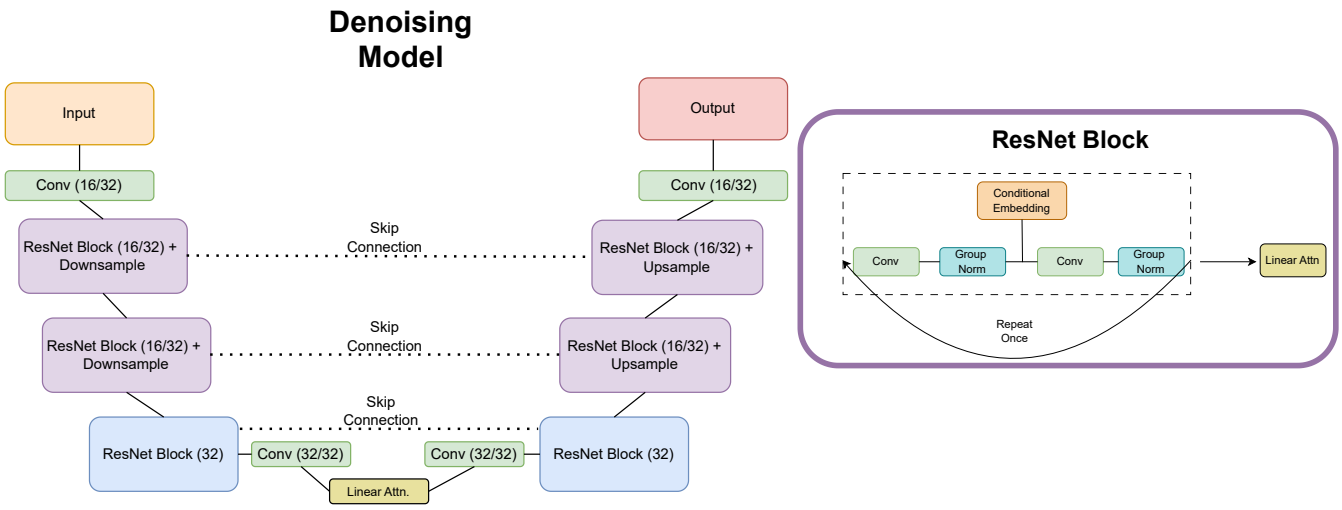

FIG. 1. Left: a schematic of the network architecture. The numbers in parentheses for each module indicate the number of filters used in that module for the network for datasets 1 and 2 / dataset 3. Right: A detailed view of the operations in a ResNet block.

contrast, the irregular binning in dataset 1 poses a challenge for fully utilizing the geometric structure of the data. Overcoming this challenge is important for the application of these techniques to real detectors, which often do not have perfectly regular geometries. Previous approaches have either used fully connected networks [16] or 1D convolutions with a very large network size [24]. Point clouds approaches have also gained some recent support as a way around this problem [30, 39].

We instead employ a new method called Geometry Latent Mapping or `GLaM`. `GLaM` learns a mapping from the data geometry to a perfectly regular geometric structure that is similar to the actual irregular geometry. This embeds the data in a regular space so that computationally efficient operations, such as cylindrical convolutions, can be used to accomplish the primary task of the ML algorithm (here, the denoising task of the diffusion model). The reverse transformation to bring the results of the primary task back to the original space is also learned by `GLaM`.

Separate mappings can be learned for different regions of the detector geometry. (A practical example is discussed below.) The embedding for a particular region is therefore only based on local information from that region. This ensures that the size of the embedding matrices remains small and that the embedded space reflects the inherent locality of the geometric structure. `GLaM` is philosophically similar to the approach of Latent Diffusion [37], which encodes data into a latent space learned by an autoencoder using a perceptual loss [50] prior to the generative task. However, with `GLaM`, the embedded space can be larger than the input space, has a direct geometric interpretation, and is learned simultaneously with the generative task. A schematic of the `GLaM` approach is shown in Fig. 2.

We apply `GLaM` to dataset 1 to learn a mapping of the input data to a regular cylindrical structure. During the diffusion training, the noise is still added in the original, irregular data structure, so that the embedding acts as just a part of the denoising model.[1] We choose the radial and angular binning of this regular structure to be the superset of all the bin boundaries of the individual layers. This results in 10 angular bins and 30 radial bins for the photon dataset and 10 angular bins and 23 radial bins for the pion dataset. A separate mapping for each layer is then learned from the original binning in that layer to this regular structure. The mapping along the radial dimension for layer ℓ is accomplished via a single matrix C^ℓ, of size $c^\ell \times c'$, where c^ℓ is the number of radial bins in the original geometry and c' is the number of bins in the regular geometry. The mapping back to the original space is likewise accomplished via a single matrix, D^ℓ, of size $c' \times c^\ell$. The values of the C^ℓ matrix are trainable parameters, but initialized to values reflecting the geometric overlap of the original and regular binning scheme:

$$C^\ell_{j,k} = \begin{cases} \frac{r_k^2 - r_{k+1}^2}{r_j^2 - r_{j+1}^2} + \kappa_{j,k} & \text{if } r_k \geq r_j \\ & \text{and } r_{k+1} \leq r_{j+1} \\ \kappa_{j,k} & \text{otherwise.} \end{cases} \tag{14}$$

Here, the values r_j denote the bin boundaries in the original geometry, r_k denote the bin boundaries in the regular geometry, and κ is a tensor of Gaussian noise with mean zero and standard deviation 10^{-5}.

[1] Adding the noise to the regular geometry results in a weak training signal for the embedding map, and therefore is more applicable to a situation in which the embedding is fixed or has been learned by some other means.

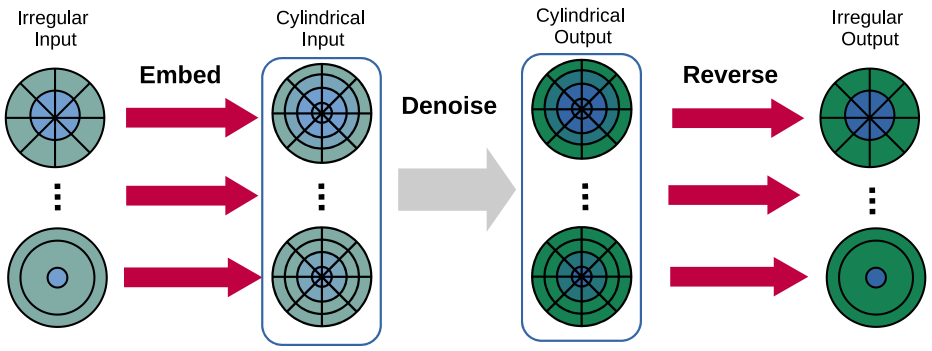

FIG. 2. A diagram of the Geometry Latent Mapping (GLaM) approach.

Because C^ℓ generally maps between spaces of different dimensionality, the matrix is rectangular and does not have an analytic inverse. D^ℓ is instead initialized to the Moore-Penrose pseudo-inverse of C^ℓ, so that initially $C^\ell D^\ell = \mathcal{I}$. However, during training it is computationally difficult to backpropagate through the Moore-Penrose pseudoinverse function, so we instead allow the values of the inverse mapping $D^\ell_{k,j}$ to be independently trainable.

In dataset 1, layers either have a single angular bin or the same 10 angular bins. We therefore take these 10 bins to be the regular structure and evenly divide the energy of layers with only a single angular bin among these 10 bins. We found that learning a mapping for the angular dimension, similar to the one used for the radial dimension, provided no performance improvements beyond the simple energy splitting.

This is quite a simple ansatz, with the embedding being fully specified by 3180 (3404) parameters for the photon (pion) dataset. We nevertheless find it works quite well in combination with cylindrical convolutions. We find that a single embedding matrix with a geometrically-informed initialization yields significantly better results than fully connected neural network layers initialized with standard techniques.

VI. RESULTS

We compare the showers generated with `CaloDiffusion` to those from `Geant4` for all datasets from the *CaloChallenge*.

A comparison of the average showers produced by `Geant4` and `CaloDiffusion`, with the `GLaM` embedding approach, for the photon sample of dataset 1 is shown in Fig. 3. A comparison of various energy distributions for the photon and pion samples of dataset 1 used in the evaluation of the *CaloChallenge* are shown in Figs. 4 and 5, respectively. The spatial properties of the shower are characterized by the Cartesian center energy of the shower, defined as $\overline{x} = \frac{\langle x_i E_i \rangle}{\sum E_i}$ for cell location x_i and energy E_i; and the shower width, defined as $\sqrt{\frac{\langle x_i^2 E_i \rangle}{\sum E_i} - \overline{x}^2}$.

For datasets 2 and 3, we examine the distribution of energy as a function of the layer of the calorimeter and as a function of the radial coordinate of the voxel. We examine the total energy of the shower, the distribution of voxel energies, and the number of voxels with energy above 1 MeV. The spatial properties of the shower are represented by the width of the shower in radial and angular dimensions (computed analogously to the Cartesian version defined above) separately for each layer of the calorimeter. The distributions for datasets 2 and 3 are shown in Figs. 6, 7, and 8.

We generally find that `CaloDiffusion` is successful at modeling all the datasets considered. The spatial distributions of the showers—the shower center and widths for dataset 1 and the layer/radial energy profile for datasets 2 and 3—are especially well reproduced. We observe only very slight degradation in quality on dataset 3 compared to dataset 2, even though it features roughly a factor of 7 higher granularity. This underscores the advantage of the convolutional approach: because it is based on fully local operations, it can readily scale to higher-dimensional data.

One of the most notable deficiencies is that `CaloDiffusion` produces a tail of low energy voxels for datasets 2 and 3, which is not seen in the `Geant4` distributions. The tail likely results from residual noise from the diffusion process that has not been fully removed by the model. The tail begins at approximately 10 keV and thus is not visible in dataset 1 because of the higher voxel energy threshold (10 MeV). The tail would likely be fully removed with a more realistic voxel minimum energy threshold applied to datasets 2 and 3. If not, such a low energy tail would still likely have minimal impact on the downstream reconstruction of the shower.

Perhaps a more relevant deficiency of the model can be seen in the distribution of the shower response, the total shower energy divided by the incident particle energy. This is seen to be particularly discrepant in the photon sample of dataset 1, in which `Geant4` exhibits a much narrower peak than `CaloDiffusion`, and mismodeling is

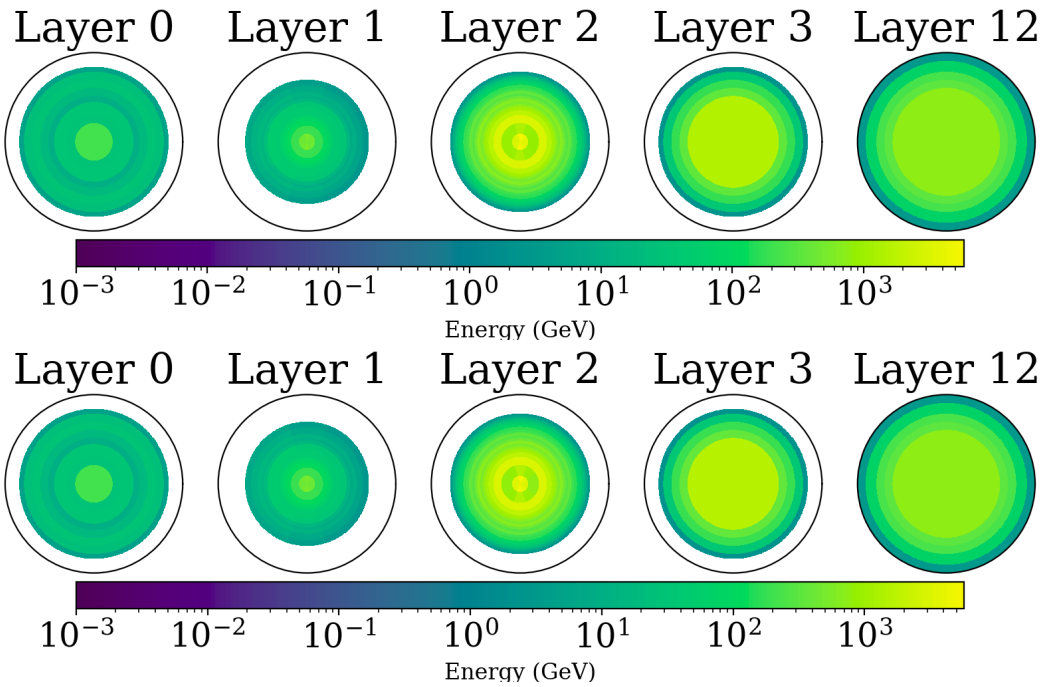

FIG. 3. A comparison of the average showers produced by `Geant4` (top) and `CaloDiffusion` (bottom) for the photon sample of dataset 1.

visible in all datasets. We have found distributions of such a 'global' property of the shower to be among the hardest for the diffusion processes to capture, because most operations are done entirely locally. For such observables, it is not straightforward to add a dedicated loss term to the diffusion training because they are only well defined at the end of the diffusion process, but most of the training uses an intermediate step[2].

In the future, a maximum mean discrepancy loss comparing the distributions from a large batch of events could be tried. Another possibility would be to adopt a two-stage generation approach, as is done in Refs. [26, 30, 31], in which the total energy of the shower, or the per-layer energy, is learned with a dedicated model and then used to normalize the output of the diffusion model.

There is also visible mismodeling of a peak in the energy distribution in layer 1 of the dataset 1 pion showers. This peak comes from very low energy pions that deposit all of their energy in layers 0 and 1 of the calorimeter, producing very sparse showers. These showers are qualitatively different from the rest and perhaps could benefit from some dedicated training or optimization.

[2] We attempted to add a dedicated L2 loss term for the total shower energy, based on estimating the final shower from the intermediate noisy shower through a 1-step estimate of the denoised shower, but it did not produce any improvements, likely because of the amount of noise in this estimate.

A. Quantitative Metrics

We compute several metrics sensitive to differences between the `Geant4` and `CaloDiffusion` samples for quantitative assessment of our model's performance.

One proposed metric [19, 51] is based on training a classifier to distinguish between the reference and synthetic samples. An optimal classifier will learn a score proportional to the likelihood ratio between the two samples. The closer the two samples are, the closer the likelihoods will be, and the classifier will struggle to distinguish between the two samples. Performance can be quantified based on the area under the curve (AUC) from the receiver-operating characteristic (ROC) curve of this classifier evaluated on a statistically independent dataset. An AUC of 1 would indicate there is a significant deficiency in the synthetic sample, such that the classifier is always able to distinguish it from a reference sample. An AUC of 0.5 would indicate the classifier cannot separate the two samples. Though Refs. [51, 52] showcased some limitations of the AUC in capturing subtle mismodelings, so far no ML-based calorimeter simulation has reported AUC scores very close to 0.5 on the *CaloChallenge* or similar datasets. Therefore, it is still a worthwhile metric to compare models.

Following the setup of the *CaloChallenge*, we employ two versions of this classifier test: one where the inputs to the classifier are the full showers themselves, along with the incident particle energy (low-level), and one where the inputs are high-level, physics-informed features of the shower (high-level). The high-level features are those

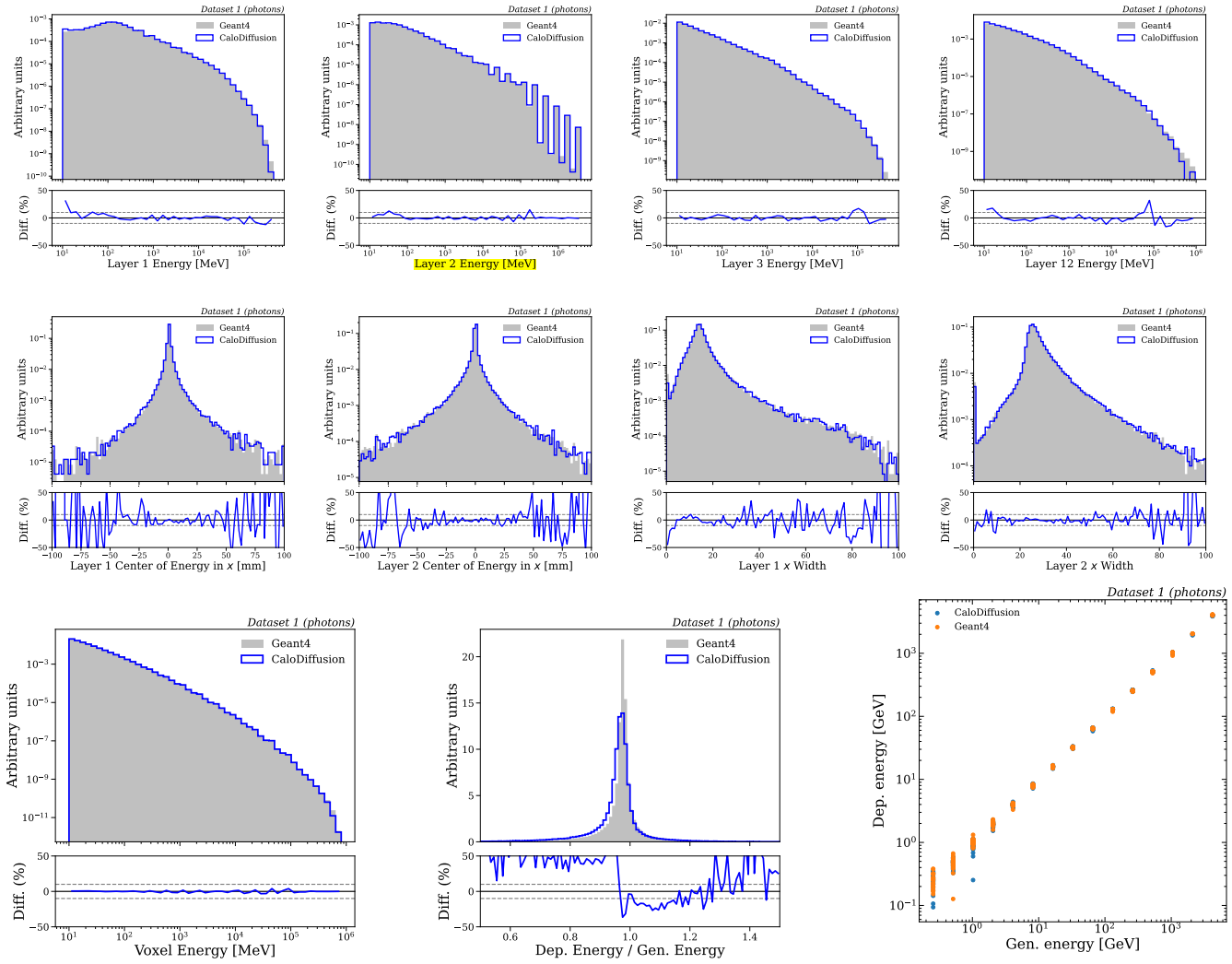

FIG. 4. A comparison between `Geant4` and `CaloDiffusion` showers across a variety of observables for the photon sample of dataset 1. The top row shows the distribution of energy in different layers of the calorimeter. The middle row shows the distribution of the center and width of the energy spread in two reference layers. The bottom row shows the distribution of voxel energies, the distribution of total shower energy divided by the incident energy, and a scatter plot of deposited energy versus incident energy.

used in the *CaloChallenge*: the incident particle energy, the energy in each layer, and the center of energy and width of the shower in the η and ϕ directions. In both cases, the classifier is a fully connected network with 2 hidden layers, each with 2048 neurons. Dropout [53], with a rate of 20%, is used after each hidden layer.

In Table I, we compare the classifier AUC values for `CaloDiffusion` to those reported by `CaloFlow`/ `iCaloFlow` [31, 54] (called here simply `CaloFlow`), and `CaloScore v2`, which are the only other models to have published quantitative results on the *CaloChallenge* at the time of writing [3]. `CaloFlow` is actually a pair of models: the originally trained 'teacher' model and a 'student' model derived from the first model, optimized for inference speed. `CaloScore v2` is a score-based diffusion model and also features distilled versions based on

[3] `CaloFlow` actually reported results for a preliminary version of the pion sample of dataset 1 without a separate evaluation sample. Therefore, those classifier AUC values were computed from the same sample of showers as used in the training. `CaloDiffusion` results are based on the final version of the pion dataset, which includes separate training and evaluation sets. When training and testing on the older sample, `CaloDiffusion` has slightly improved AUC values, but we report here results on the final *CaloChallenge* version for posterity.

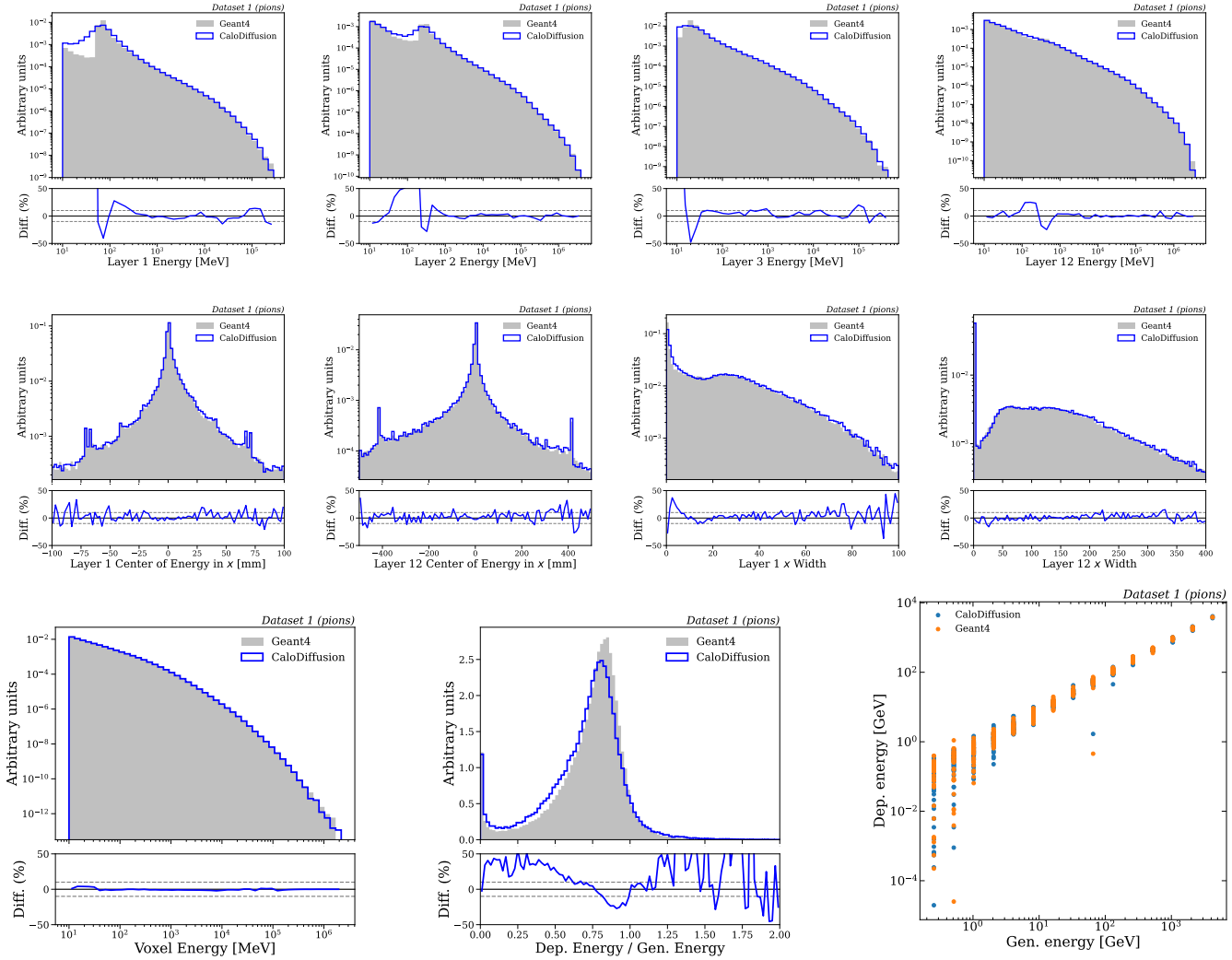

FIG. 5. A comparison between `Geant4` and `CaloDiffusion` showers across a variety of observables for the pion sample of dataset 1. The top row shows the distribution of energy in different layers of the calorimeter. The middle row shows the distribution of the center and width of the energy spread in two reference layers. The bottom row shows the distribution of voxel energies, the distribution of total shower energy divided by the incident energy, and a scatter plot of deposited energy versus incident energy.

progressive distillation. `CaloScore v2` did not provide results for the pion version of dataset 1. As this version of `CaloDiffusion` was optimized for sample quality and has not used dedicated methods to improve sampling time, we compare to the teacher model of `CaloFlow` and the undistilled version of `CaloScore v2`[4], which have better performance and a similar generation time to `CaloDiffusion`. Future work will explore the development of a new version of `CaloDiffusion` with optimized generation speed, which would be more suitable for comparison to the faster versions of each model.

[4] For dataset 3, the `CaloScore v2` authors do not provide results on a model without distillation, so we compare to the 8-step distilled version.

We find that `CaloDiffusion` produces classifier AUC values below 0.7 for all four datasets, indicating that the classifier struggles to distinguish between `CaloDiffusion` and `Geant4` showers. `CaloDiffusion` achieves better AUC scores than `CaloFlow` and `CaloScore v2` for all cases except the photon showers of dataset 1 when using high-level features. The performance gains of `CaloDiffusion` are especially prominent for the higher-dimensional datasets 2 and 3.

For `CaloDiffusion`, the classifiers trained on low-level features and high-level features have quite similar AUC values. This indicates that most of the discrimination power between `CaloDiffusion` and `Geant4` showers is captured by these high-level features. We generally find that the low-level classifier overfits the training set sig-

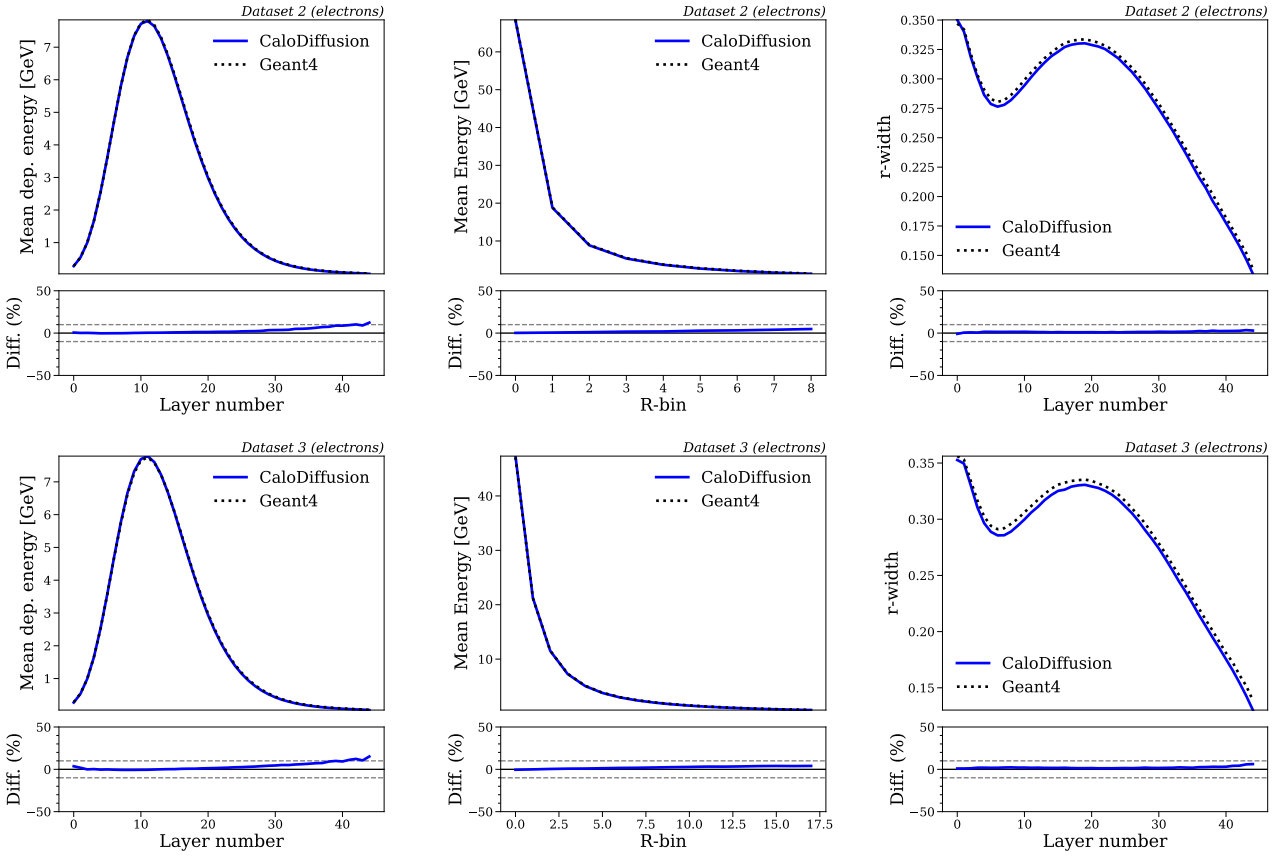

FIG. 6. A comparison between `Geant4` and `CaloDiffusion` showers on datasets 2 (top row) and 3 (bottom row). The average shower energy is shown as a function of layer (left) and as a function of radial bin (center). The width of the shower in the radial direction is also shown (right).

	Classifier AUC (low / high)		
Dataset	`CaloDiffusion`	`CaloFlow`	`CaloScore v2`
1 (photons)	**0.62** / 0.62	0.70 / **0.55**	0.76 / 0.59
1 (pions)	**0.65** / **0.65**	0.78 / 0.70	- / -
2 (electrons)	**0.56** / **0.56**	0.80 / 0.80	0.60 / 0.62
3 (electrons)	**0.56** / **0.57**	0.91 / 0.95	0.67 / 0.85

TABLE I. The AUC values for a classifier trained to distinguish between `Geant4` and synthetic showers. The first value listed is the AUC for the classifier trained on low-level features and the second is the AUC for the classifier trained on high-level features. The `CaloDiffusion` values are the average of 5 independent classifier trainings. In all cases, the variation in scores was observed to be 0.01 or less. In each row, the bold value is the best AUC value for each classifier type.

Dataset	FPD	KPD
1 (photons)	0.014(1)	0.004(1)
1 (pions)	0.029(1)	0.004(1)
2 (electrons)	0.043(2)	0.0001(2)
3 (electrons)	0.031(2)	0.0001(1)

TABLE II. Additional metrics comparing the agreement between showers generated with `Geant4` and `CaloDiffusion`. The number in parentheses is the uncertainty in the last significant digit as evaluated with the `JETNET` library.

nificantly, and therefore an improved architecture would perhaps perform better. However, we generally take this overfitting to be a positive sign, because it indicates that distinguishing between `Geant4` and `CaloDiffusion` showers based on generalizable features is not easy.

We additionally report the Fréchet Particle Distance (FPD) and Kernel Particle Distance (KPD) metrics, suggested in Ref. [52] and implemented in the `JETNET` library [55], interfaced to the *CaloChallenge* evaluation code. We use the same high-level shower features as in the classifier test but omit the incident particle energy. We find that the FPD metric computed with these features is slightly biased; the reported value does not agree with zero within its uncertainty, even when comparing two samples of `Geant4` showers. We therefore normalize our reported values for FPD by subtracting the value

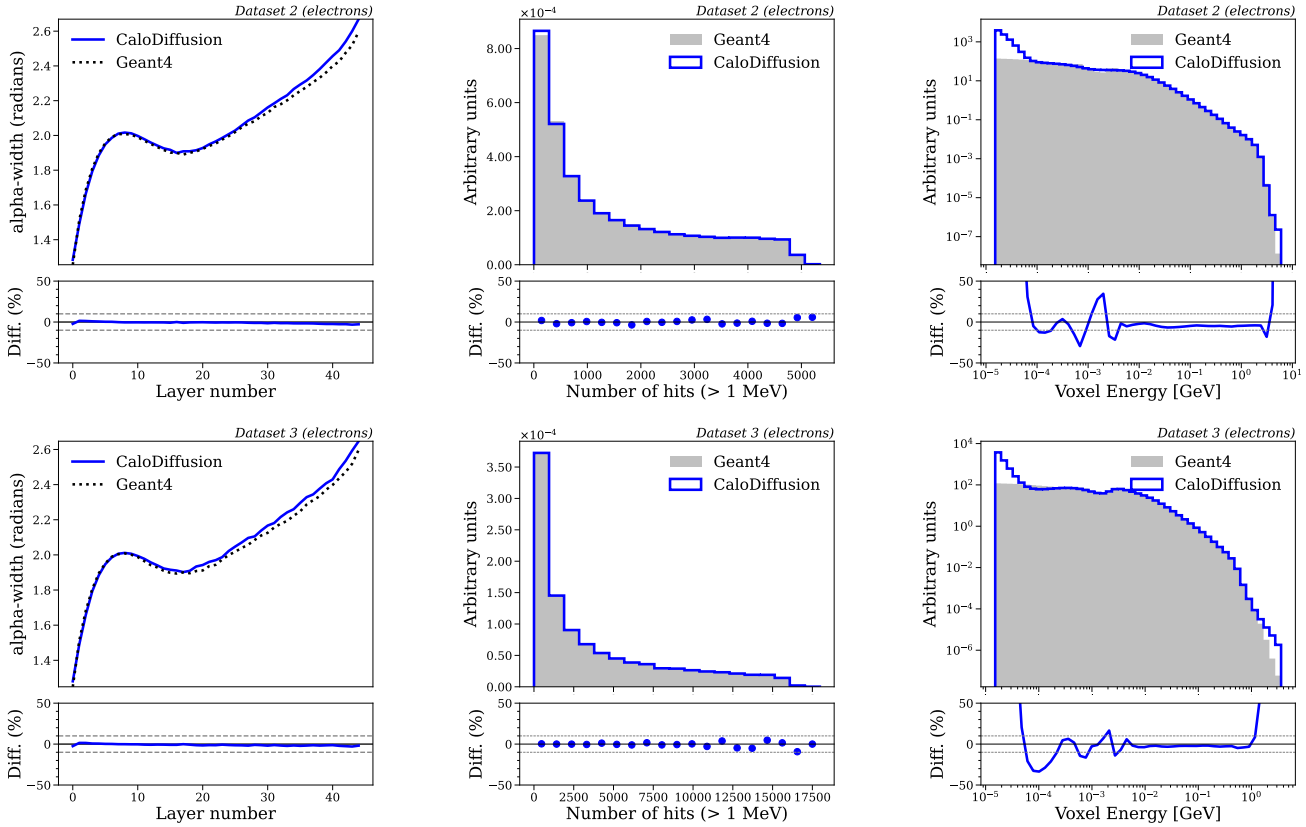

FIG. 7. A comparison between `Geant4` and `CaloDiffusion` showers on datasets 2 (top row) and 3 (bottom row). The quantities shown are the width of the shower in the angular direction (left), the distribution of total number of non-zero voxels (center), and the energy per voxel (right).

computed comparing two `Geant4` samples[5]. We report these additional metrics in Table II.

Further quantitative comparisons with other approaches will be performed at the conclusion of the *CaloChallenge*. However, initial results from the *CaloChallenge* [56] indicated that a preliminary version of `CaloDiffusion`[6] was among the top submissions for every dataset.

Ablation studies quantifying the performance improvements for various aspects of `CaloDiffusion` are discussed in Appendix A.

B. Timing

In Table III, we report the generation time of our model using different batch sizes on both CPUs and GPUs. Results are based on a 2.6 GHz Intel E5-2650v2 "Ivy Bridge" 8-Core CPU and an NVIDIA V100 GPU. The time required to generate a shower in `Geant4` depends strongly on the incident energy of the particle. The average over the incident energies used in datasets 2 and 3 is $O(100\text{ s})$ [31].

Dataset	Batch Size	Time/Shower [s] CPU	GPU
1 (photons)	1	9.4	6.3
(368 voxels)	10	2.0	0.6
	100	1.0	0.1
1 (pions)	1	9.8	6.4
(533 voxels)	10	2.0	0.6
	100	1.0	0.1
2 (electrons)	1	14.8	6.2
(6.5K voxels)	10	4.6	0.6
	100	4.0	0.2
3 (electrons)	1	52.7	7.1
(40.5K voxels)	10	44.1	2.6
	100	-	2.0

TABLE III. The shower generation time for `CaloDiffusion` on CPU and GPU for various batch sizes.

Because of the iterative denoising process during generation, diffusion models are usually slower than other

[5] The FPD values computed comparing two `Geant4` samples are 0.008, 0.0005, 0.008, and 0.011 for datasets 1 (photons), 1 (pions), 2 (electrons), and 3 (electrons), respectively.

[6] The preliminary version did not use the attention layers and dimensionality reduction in z that are included in the U-net architecture of the version in this paper.

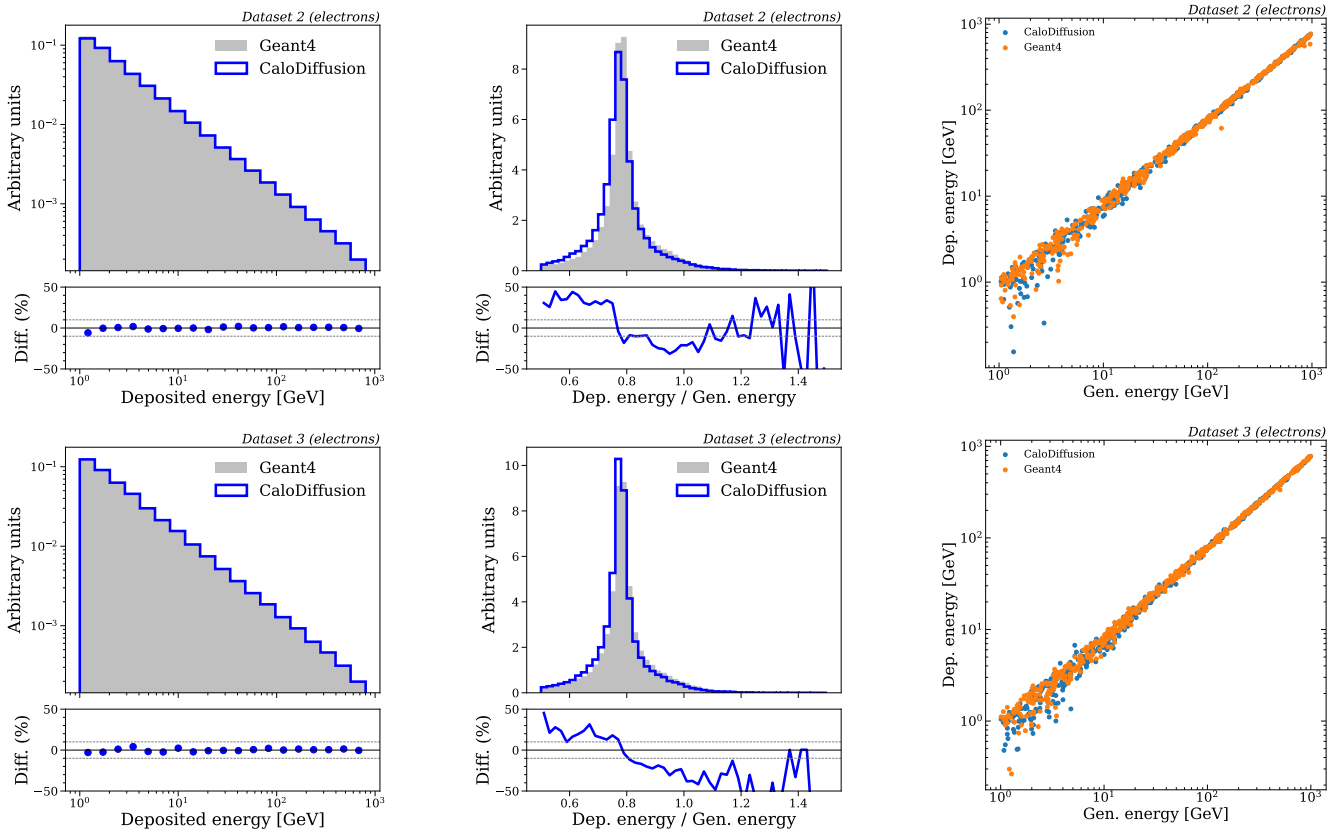

FIG. 8. A comparison between `Geant4` and `CaloDiffusion` showers on datasets 2 (top row) and 3 (bottom row). Shown are the distributions of the total shower energy (left) and the total shower energy divided by the incident particle energy (center), and a scatter plot of the two quantities (right).

ML approaches. If limited to a batch size of one and running on a CPU, this version of `CaloDiffusion` may not satisfy the computation time requirement for a fast simulation. Without any additional training or algorithmic changes, the `CaloDiffusion` generation time can be linearly improved by reducing the number of diffusion steps used in the sampling, with a cost to sample quality. We explore this tradeoff in Section VI C.

C. Sampling Steps and Quality

In this work, our main goal was to demonstrate the fidelity achievable with the `CaloDiffusion` approach, rather than optimizing for generation speed. We therefore chose the fewest diffusion steps that did not exhibit a significant decrease in sample quality. However, significant reductions in the number of sampling steps can still result in high-quality samples. This tradeoff between number of diffusion steps and sample quality was studied using dataset 2. By changing the noise schedule, the model can be sampled using different numbers diffusion steps without retraining. Inference time scales linearly with the number of diffusion steps regardless of batch size (using 200 steps generates samples twice as fast as 400 steps). We find that one of the distributions most sensitive to the number of diffusion steps is the ratio of deposited to incident energy. This seems to be one of the hardest features for the diffusion model to capture, and it degrades further with fewer steps. A plot of this feature with different numbers of diffusion steps is shown in Fig. 9. In addition to the metrics reported in Sec. VI A, we report the separation power between `Geant4` and `CaloDiffusion` on this 1D distribution. The separation power is a modified χ^2 metric proposed for calorimeter simulation in Ref. [18] and implemented in the *CaloChallenge* framework. Results are presented in Table IV.

Num. Steps	Classifier AUC (low / high)	FPD	E Ratio Sep. Power
400	0.56 / 0.55	0.043(1)	0.011
200	0.61 / 0.56	0.046(1)	0.036
100	0.69 / 0.59	0.065(3)	0.079
50	0.83 / 0.67	0.110(4)	0.251

TABLE IV. Quantitative metrics comparing the agreement between showers generated with `Geant4` and `CaloDiffusion` with different numbers of sampling steps for dataset 2 of the *CaloChallenge*. The separation power is computed using the ratio of deposited to incident energy. See text for details.

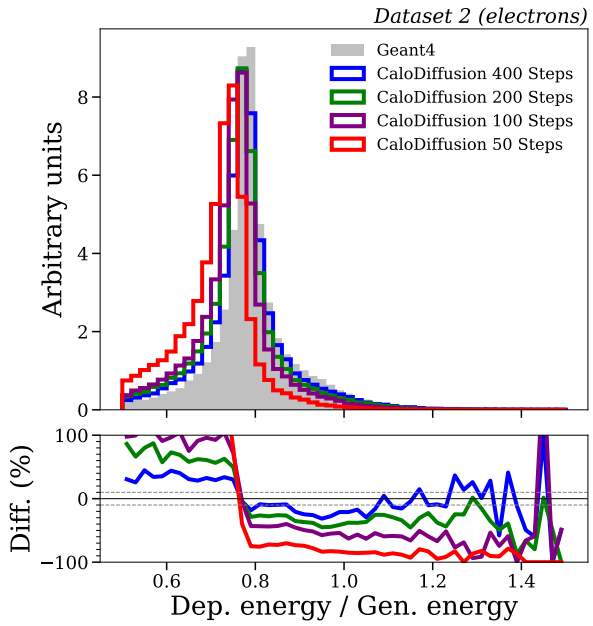

FIG. 9. Distribution of the ratio of incident particle energy and total deposited energy of the shower comparing `CaloDiffusion` samples generated with different numbers of sampling steps to `Geant4`.

Improving the generation time of diffusion models is an active area of research in the machine learning community. Improved sampling algorithms have been proposed and shown to achieve higher sample quality for low numbers of diffusion steps [38]. Alternatively, once trained, the diffusion model can be 'distilled' into a new model which requires an order of magnitude fewer diffusion steps [57, 58] with minimal loss in sample quality. This distillation approach was recently employed for the generation of particle jets using a point cloud representation in Refs. [29, 40] and for detector simulation in Ref. [32], still with some loss of quality.

An alternative approach would be to simplify the diffusion task of the network. The dimensionality of the data can be reduced by first compressing to a smaller latent space, running diffusion, and then decompressing back to the original space [37]. Alternatively, rather than starting the diffusion process from pure noise, it has been demonstrated that diffusion between two images is possible [59]. One could therefore start the diffusion process from an approximate calorimeter simulation, generated by current non-ML fast simulation techniques. By providing input similar to the final result, the diffusion process would likely require fewer steps and some physical features may be learned more easily. This would be a similar approach to Refs. [60], in which CNNs were used to denoise a fast simulation to achieve higher quality results. A conceptually related approach using diffusion with a Schrödinger bridge was recently demonstrated [61]. These techniques for refinement of low-level hits in calorimeter showers can complement regression-based refinement of high-level observables [62] by making the latter easier to learn and therefore even more precise.

VII. CONCLUSION

In this work, we introduced `CaloDiffusion`, a new machine learning (ML) model that uses diffusion to generate calorimeter showers. We employed several novel optimizations that exploit the underlying geometry of the calorimeter data. We have also introduced the geometry latent mapping (`GLaM`), a new approach to handle irregular geometrical structures in data. `GLaM` learns a lightweight embedding to transform the irregular data geometry into a regular shape, which can then be used in symmetry-preserving operations such as convolutions, and also learns the reverse transformation. We have demonstrated that `CaloDiffusion`, combined with `GLaM`, is able to generate high quality showers on a variety of datasets, some with high dimensionality. We have set new benchmarks in quantitative performance metrics that demonstrate it is difficult to distinguish between `CaloDiffusion` and `Geant4` showers.

Our work significantly advances the state of the art in the achievable physics performance from ML-based fast simulation techniques. This is an important step to establish the viability of such techniques to resolve the simulation component of the computing challenges in the High Luminosity LHC era. While the unoptimized generation time for diffusion models is slower than for some other ML architectures, producing showers in batches on GPUs is already noticeably faster than the `Geant4`-based full detector simulation. Future work will explore and compare a variety of approaches to improve the generation speed of `CaloDiffusion` and will apply `CaloDiffusion` with `GLaM` to datasets with even more complicated geometries.

CODE AVAILABILITY

The code to reproduce the results in this paper, as well as the trained models, can be found at https://github.com/OzAmram/CaloDiffusionPaper.

ACKNOWLEDGMENTS

We thank the organizers of the *CaloChallenge* for providing the community datasets and evaluation code used in this work. We thank Raghav Kansal for assistance computing the KPD/FPD metrics on the *CaloChallenge* datasets.

FUNDING INFORMATION

O. Amram and K. Pedro are supported by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. O. Amram is supported by the U.S. CMS Software and Computing Operations Program under the U.S. CMS HL-LHC R&D Initiative.

[1] **GEANT4** Collaboration, S. Agostinelli et al., *GEANT4–a simulation toolkit*, *Nucl. Instrum. Meth. A* **506** (2003) 250.

[2] J. Allison et al., *Geant4 developments and applications*, *IEEE Trans. Nucl. Sci.* **53** (2006) 270.

[3] J. Allison et al., *Recent developments in Geant4*, *Nucl. Instrum. Meth. A* **835** (2016) 186.

[4] **HEP Software Foundation** Collaboration, J. Apostolakis et al., *HEP Software Foundation Community White Paper Working Group - Detector Simulation*, arXiv:1803.04165.

[5] **CMS** Collaboration, *The Phase-2 upgrade of the CMS endcap calorimeter*, CMS Technical Design Report CERN-LHCC-2017-023, CMS-TDR-019, 2017. https://cds.cern.ch/record/2293646.

[6] **CMS** Collaboration, K. Pedro, *Integration and Performance of New Technologies in the CMS Simulation*, *EPJ Web Conf.* **245** (2020) 02020, [arXiv:2004.02327].

[7] X. Ju et al., *Performance of a geometric deep learning pipeline for HL-LHC particle tracking*, *Eur. Phys. J. C* **81** (2021) 876, [arXiv:2103.06995].

[8] S. Abdullin, P. Azzi, F. Beaudette, P. Janot, and A. Perrotta, *The fast simulation of the CMS detector at LHC*, *J. Phys. Conf. Ser.* **331** (2011) 032049.

[9] A. Giammanco, *The Fast Simulation of the CMS Experiment*, *J. Phys. Conf. Ser.* **513** (2014) 022012.

[10] **CMS** Collaboration, S. Sekmen, *Recent Developments in CMS Fast Simulation*, *PoS* **ICHEP2016** (2016) 181, [arXiv:1701.03850].

[11] **ATLAS** Collaboration, M. Beckingham, M. Duehrssen, E. Schmidt, M. Shapiro, M. Venturi, J. Virzi, I. Vivarelli, M. Werner, S. Yamamoto, and T. Yamanaka, *The simulation principle and performance of the ATLAS fast calorimeter simulation FastCaloSim*, ATLAS PUB Note, CERN, Geneva, 10, 2010. http://cds.cern.ch/record/1300517.

[12] W. Lukas, *Fast Simulation for ATLAS: Atlfast-II and ISF*, *J. Phys. Conf. Ser.* **396** (2012) 022031.

[13] **ATLAS** Collaboration, *AtlFast3: the next generation of fast simulation in ATLAS*, *Comput. Softw. Big Sci.* **6** (2022) 7, [arXiv:2109.02551].

[14] M. Paganini, L. de Oliveira, and B. Nachman, *CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks*, *Phys. Rev. D* **97** (2018) 014021, [arXiv:1712.10321].

[15] V. Chekalina, E. Orlova, F. Ratnikov, D. Ulyanov, A. Ustyuzhanin, and E. Zakharov, *Generative Models for Fast Calorimeter Simulation: the LHCb case*, *EPJ Web Conf.* **214** (2019) 02034, [arXiv:1812.01319].

[16] **ATLAS** Collaboration, *Fast simulation of the ATLAS calorimeter system with Generative Adversarial Networks*, ATLAS PUB Note, CERN, Geneva, 2020. https://cds.cern.ch/record/2746032.

[17] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, and K. Krüger, *Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed*, *Comput. Softw. Big Sci.* **5** (2021) 13, [arXiv:2005.05334].

[18] S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman, and D. Shih, *DCTRGAN: Improving the Precision of Generative Models with Reweighting*, *JINST* **15** (2020) P11004, [arXiv:2009.03796].

[19] C. Krause and D. Shih, *Fast and accurate simulations of calorimeter showers with normalizing flows*, *Phys. Rev. D* **107** (2023) 113003, [arXiv:2106.05285].

[20] C. Krause and D. Shih, *Accelerating accurate simulations of calorimeter showers with normalizing flows and probability density distillation*, *Phys. Rev. D* **107** (2023) 113004, [arXiv:2110.11377].

[21] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, and K. Krüger, *Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network*, *EPJ Web Conf.* **251** (2021) 03003, [arXiv:2102.12491].

[22] **ATLAS** Collaboration, *Deep generative models for fast photon shower simulation in ATLAS*, arXiv:2210.06204.

[23] E. Buhmann, S. Diefenbacher, D. Hundhausen, G. Kasieczka, W. Korcari, E. Eren, F. Gaede, K. Krüger, P. McKeown, and L. Rustige, *Hadrons, better, faster, stronger*, *Mach. Learn. Sci. Tech.* **3** (2022) 025014, [arXiv:2112.09709].

[24] V. Mikuni and B. Nachman, *Score-based generative models for calorimeter shower simulation*, *Phys. Rev. D* **106** (2022) 092009, [arXiv:2206.11898].

[25] E. Buhmann, G. Kasieczka, and J. Thaler, *EPiC-GAN: Equivariant Point Cloud Generation for Particle Jets*, arXiv:2301.08128.

[26] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh, and D. Shih, *L2LFlows: Generating High-Fidelity 3D Calorimeter Images*, arXiv:2302.11594.

[27] H. Hashemi, N. Hartmann, S. Sharifzadeh, J. Kahn, and T. Kuhr, *Ultra-High-Resolution Detector Simulation with Intra-Event Aware GAN and Self-Supervised Relational Reasoning*, arXiv:2303.08046.

[28] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, K. Krüger, P. McKeown, and L. Rustige, *New Angles on Fast Calorimeter Shower Simulation*, arXiv:2303.18150.

[29] V. Mikuni, B. Nachman, and M. Pettee, *Fast Point Cloud Generation with Diffusion Models in High Energy Physics*, arXiv:2304.01266.

[30] 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, arXiv:2305.04847.

[31] M. R. Buckley, C. Krause, I. Pang, and D. Shih, *Inductive CaloFlow*, arXiv:2305.11934.

[32] V. Mikuni and B. Nachman, *CaloScore v2: Single-shot Calorimeter Shower Simulation with Diffusion Models*, arXiv:2308.03847.

[33] A. Adelmann et al., *New directions for surrogate models and differentiable programming for High Energy Physics detector simulation*, in *Snowmass 2021*, 3, 2022. arXiv:2203.08806.

[34] S. Badger et al., *Machine learning and LHC event generation*, *SciPost Phys.* **14** (2023) 079, [arXiv:2203.07460].

[35] M. Barbetti, *Lamarr: LHCb ultra-fast simulation based on machine learning models deployed within Gauss*, in *21th International Workshop on Advanced Computing and Analysis Techniques in Physics Research: AI meets Reality*, 3, 2023. arXiv:2303.11428.

[36] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, p. 6840, Curran Associates, Inc., 2020. arXiv:2006.11239.

[37] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 10684, June, 2022. arXiv:2112.10752.

[38] T. Karras, M. Aittala, T. Aila, and S. Laine, *Elucidating the design space of diffusion-based generative models*, arXiv:2206.00364.

[39] F. T. Acosta, V. Mikuni, B. Nachman, M. Arratia, K. Barish, B. Karki, R. Milton, P. Karande, and A. Angerami, *Comparison of Point Cloud and Image-based Models for Calorimeter Fast Simulation*, arXiv:2307.04780.

[40] M. Leigh, D. Sengupta, J. A. Raine, G. Quétant, and T. Golling, *PC-Droid: Faster diffusion and improved quality for particle cloud generation*, arXiv:2307.06836.

[41] A. Shmakov, K. Greif, M. Fenton, A. Ghosh, P. Baldi, and D. Whiteson, *End-To-End Latent Variational Diffusion Models for Inverse Problems in High Energy Physics*, arXiv:2305.10399.

[42] A. Butter, N. Huetsch, S. P. Schweitzer, T. Plehn, P. Sorrenson, and J. Spinner, *Jet Diffusion versus JetGPT – Modern Networks for the LHC*, arXiv:2305.10475.

[43] V. Mikuni and B. Nachman, *High-dimensional and Permutation Invariant Anomaly Detection*, arXiv:2306.03933.

[44] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska, *Fast Calorimeter Simulation Challenge*, 2022. https://calochallenge.github.io/homepage/.

[45] **ATLAS** Collaboration, *Datasets used to train the Generative Adversarial Networks used in ATLFast3*, 2021. https://doi.org/10.7483/OPENDATA.ATLAS.UXKX.TXBN.

[46] A. Q. Nichol and P. Dhariwal, *Improved denoising diffusion probabilistic models*, in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, p. 8162, PMLR, 18–24 Jul, 2021. arXiv:2102.09672.

[47] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), p. 234, Springer International Publishing, 2015. arXiv:1505.04597.

[48] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 770, 2016. arXiv:1512.03385.

[49] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*, arXiv:2006.16236.

[50] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, *The unreasonable effectiveness of deep features as a perceptual metric*, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), p. 586, IEEE Computer Society, 6, 2018. arXiv:1801.03924.

[51] R. Das, L. Favaro, T. Heimel, C. Krause, T. Plehn, and D. Shih, *How to Understand Limitations of Generative Networks*, arXiv:2305.16774.

[52] R. Kansal, A. Li, J. Duarte, N. Chernyavskaya, M. Pierini, B. Orzari, and T. Tomei, *Evaluating generative models in high energy physics*, *Phys. Rev. D* **107** (2023) 076017, [arXiv:2211.10295].

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

[54] C. Krause, I. Pang, and D. Shih, *CaloFlow for CaloChallenge Dataset 1*, arXiv:2210.14245.

[55] R. Kansal, J. Duarte, C. Pareja, L. Action, Z. Hao, and mova, *jet-net/JetNet: v0.2.3.post3*, 3, 2023. https://doi.org/10.5281/zenodo.7778868.

[56] C. Krause, *The Fast Calorimeter Challenge 2022: Results and The Road Ahead*, 5, 2023. CaloChallenge Workshop, https://agenda.infn.it/event/34036/contributions/200888/attachments/106010/149192/CaloChallenge.Summary.C.Krause.pdf.

[57] T. Salimans and J. Ho, *Progressive distillation for fast sampling of diffusion models*, in *International Conference on Learning Representations*, 2022. arXiv:2202.00512.

[58] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, *Consistency models*, arXiv:2303.01469.

[59] A. Bansal, E. Borgnia, H.-M. Chu, J. S. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein, *Cold diffusion: Inverting arbitrary image transforms without noise*, arXiv:2208.09392.

[60] S. Banerjee, B. C. Rodriguez, L. Franklin, H. G. De La Cruz, T. Leininger, S. Norberg, K. Pedro, A. Rosado Trinidad, and Y. Ye, *Denoising Convolutional Networks to Accelerate Detector Simulation*, *J. Phys. Conf. Ser.* **2438** (2023) 012079, [arXiv:2202.05320].

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

[62] S. Bein, P. Connor, K. Pedro, P. Schleper, and M. Wolf, *Refining fast simulation using machine learning*, in *26th International Conference on Computing in High Energy & Nuclear Physics*, 9, 2023. arXiv:2309.12919.

Appendix A: Ablation Studies

We report here ablation studies for several of the innovations and design choices used in this study. We perform this ablation study on the pion sample of dataset 1 because it is the most difficult sample for `CaloDiffusion` to reproduce.

The ablations we consider are:

- Not using the 'layer' and 'radial' images that allow for location-conditional convolutions.
- Using regular Cartesian convolutions instead of cylindrical ones.
- Using a fixed geometric embedding instead of the learnable `GLaM` approach. The setup used to initialize `GLaM` (Section V B), based on the area overlap of cells, is employed for the fixed embedding.

We attempted an additional ablation that replaced `GLaM` with several fully connected dense layers, using no geometric information, but this model did not produce any reasonable results at the denoising task.

For each choice under study, we retrained a different version of `CaloDiffusion` and generate samples to evaluate the impact. For the definitions of the metrics reported, see Tables I and II. We find that each ablation does lead to worse performance than the baseline, but no single change results in a substantial drop in performance.

Model	Classifier AUC (low / high)	FPD	E Ratio Sep. Power
Baseline Model	0.65 / 0.65	0.029(1)	0.0093
Without layer and radial images	0.67 / 0.69	0.038(1)	0.0120
Without cylindrical convolutions	0.67 / 0.69	0.035(1)	0.0110
Fixed geometric embedding	0.66 / 0.69	0.039(2)	0.0118

TABLE V. Quantitative metrics comparing the agreement between showers generated with `Geant4` and different ablations of `CaloDiffusion` for the pion sample of dataset 1 of the *CaloChallenge*.