CaloDREAM — Detector Response Emulation via Attentive flow Matching

Luigi Favaro[1,2], Ayodele Ore[1], Sofia Palacios Schweitzer[1], and Tilman Plehn[1]

1 Institut für Theoretische Physik, Universität Heidelberg, Germany
2 CP3, Université catholique de Louvain, Louvain-la-Neuve, Belgium

February 11, 2025

Abstract

Detector simulations are an exciting application of modern generative networks. Their sparse high-dimensional data combined with the required precision poses a serious challenge. We show how combining Conditional Flow Matching with transformer elements allows us to simulate the detector phase space reliably. Namely, we use an autoregressive transformer to simulate the energy of each layer, and a vision transformer for the high-dimensional voxel distributions. We show how dimension reduction via latent diffusion allows us to train more efficiently and how diffusion networks can be evaluated faster with bespoke solvers. We showcase our framework, CaloDREAM, on datasets 2 and 3 of the CaloChallenge.

Contents

---

1 Introduction

Simulations are the way we compare theory predictions to LHC data, allowing us to draw conclusions about fundamental theory from complex scattering data [1,2]. The modular simulation chain starts from the hard interaction and progresses through particle decays, QCD radiation, hadronization, hadron decays, to the interaction of all particles with the detector. Currently, the last step is turning into a bottleneck in speed and precision. Generating calorimeter showers with GEANT4 [3–5], based on first principles, requires a large fraction of the computing budget. Without significant progress, this simulation step will be the limiting factor for all analyses at the HL-LHC [6,7].

Modern machine learning is transforming the way we simulate LHC data [8]. In the past few years we have seen successful applications to all steps in the simulation chain [2], phase space integration [9–19]; parton showers [20–27]; hadronization [28–31]; detector simulations [32–65], and end-to-end event generation [66–73], including inverse simulations [74–84] and simulation-based inference [85–87]. While these new concepts and tools have the potential to transform LHC simulations, we need to ensure that these networks and their technical strengths can be understood. This is the only way that we can systematically improve the LHC simulation chain [71,88–91], without endangering the key role it plays in, essentially, every LHC analysis. Most notably, we must avoid the case where effects of interest are absorbed into LHC simulations as a result of data-driven modelling. This means that for now we always assume that networks used in LHC simulations are trained on simulations and controlled by comparing to simulations.

In this paper, we will apply cutting-edge generative networks to calorimeter shower simulations. The high-dimensional phase spaces of calorimeter showers are a challenge to the established normalizing flows or INNs [92], and different variants of diffusion networks appear to be the better-suited architecture [58]. This is in spite of the fact that diffusion networks are, typically, slower in the forward generation and do not allow for an efficient likelihood extraction. In addition to showing that these networks are able to simulate sparse phase space signals like calorimeter showers, we will explore which phase space dimensionalities we can describe with full-dimensional latent spaces and how a dimension-reduced latent representation affects the network performance.

Given the GEANT4 benchmark presented in Sec. 2, we will see that a factorized approach is most promising. In Sec. 3 we first introduce a Conditional Flow Matching (CFM) network combined with an autoregressive transformer to learn the layer energies. Next, we combine it with a 3-dimensional vision transformer to learn the shower shapes. This combination can be trained on datasets 2 and 3 of the CaloChallenge to generate high-fidelity calorimeter showers. In this application the step from dataset 2 to dataset 3 motivates a switch from full-dimensional voxel representations to a dimension-reduced latent space [50]. In Sec. 4 we study, in some detail, how the full-dimension generative network encodes the calorimeter shower information for both datasets. To alleviate the computational challenges, we also show how a lower-dimensional latent representation helps us describe high-dimensional data like the Calo Challenge dataset 3 and how the CFM networks can sample more efficiently. For quantitative benchmarking of the learned phase space distribution we employ a learned classifier test, indicating that the network precision for both datasets is at the per-cent level and the loss in precision from a reduced latent space is controlled, including its only failure mode, which are the sparsity distributions.

2 Data and preprocessing

To benchmark our new network architectures, we use dataset 2 (DS2) [93] and dataset 3 (DS3) [94] of the CaloChallenge 2022 [95]. Each set consists of 200k GEANT4 [3] electron showers: 100k for training/validation and 100k for testing. Showers are simulated over a log-uniform incident energy range

$$E_{\text{inc}} = 10^3 \ldots 10^6 \text{ MeV} . \tag{1}$$

The physical detector has a cylindrical geometry with alternating layers of absorber and active material, altogether 90 layers. The voxelization following Ref. [96] combines an active layer and an absorber layer resulting in 45 concentric cylindrical layers.

The particle originating the shower always enters at the (0,0,0) location and defines the z-axis of the coordinate system. The number of readout cells per layer is defined in a polar coordinate system and it is different for DS2 and DS3. DS2 has a total of 6480 voxels: 144 voxels per layer, each divided into 16 angular and 9 radial bins. DS3 has a much higher granularity with 40500 total voxels, where the number of layers is unchanged but the angular and radial binning is 50×18. Both datasets have a threshold of 15.15 keV. While this is an unrealistic cut for practical applications, it provides a useful challenge to high-dimensional generative networks covering a wide energy range.

Preprocessing

We improve our training by including a series of preprocessing steps, similar to previous studies [49,51,57,58,92]. We split information on the deposited energy from its distribution over voxels by introducing energy ratios [41]

$$u_0 = \frac{\sum_i E_i}{f\, E_{\text{inc}}} \qquad \text{and} \qquad u_i = \frac{E_i}{\sum_{j\geq i} E_j} \quad i = 1, \dots, 44 \,, \tag{2}$$

where E_i refers to the total energy deposited in layer i, and $f \in \mathbb{R}$ is a scale factor. The number of u-variables matches the number of layers. With these variables extracted from a given shower, we are free to normalize the voxel values by the energy of their corresponding layer without losing any information. This definition is analytically invertible, imposes energy conservation, and ensures that the normalized voxels and each $u_{i>0}$ are always in the range $[0, 1]$. However, due to the calibration of the detector response caused by the inactive material, u_0 can have values larger than 1. We set $f = 2.85$ in Eq.(2), to rescale $u_0 \in [0, 1]$. All networks are conditioned on E_{inc}. This quantity is passed to the network after a log transformation and a rescaling into the unit interval.

To train the autoencoders used for dimensionality reduction we do not use any additional preprocessing steps. For the setup using the full input space, we apply a logit transformation regularized by the parameter α which rescales each input voxel x,

$$\begin{aligned} x_\alpha &= (1-2\alpha)x + \alpha \in [\alpha, 1-\alpha] \qquad \text{with} \quad \alpha = 10^{-6} \\ x' &= \log \frac{x_\alpha}{1-x_\alpha} \,. \end{aligned} \tag{3}$$

Finally, we calculate the mean and the standard deviation of the training dataset and standardize each feature. The postprocessing includes an additional step that rescales the sum of the generated voxels to ensure the correct normalization in each layer.

3 CaloDREAM

In CaloDREAM[*], we employ two generative networks, one energy network and one shape network [41]. The energy network learns the energy-ratio features conditioned on the incident energy, $p(u_i|E_{\text{inc}})$. The shape network learns the conditional distribution for the voxels, $p(x|E_{\text{inc}}, u)$. The two networks are trained independently, but are linked in the generative process. Specifically, to sample showers given an incident energy, we follow the chain

$$\begin{aligned} u_i &\sim p_\phi(u_i|E_{\text{inc}}) \\ x &\sim p_\theta(x|E_{\text{inc}}, u) \,. \end{aligned} \tag{4}$$

In this notation ϕ stands for the weights in the energy network and θ for the weights in the shape network. Although the number of calorimeter layers is consistent across DS2 and DS3 and the underlying showers are the same, we train separate energy networks for each dataset. The incident energy is always sampled from the known distribution in the datasets, as in Eq.(1).

3.1 Energy network — Transfusion

Both of our generative networks use the Conditional Flow Matching architecture [97]. It starts with the ordinary differential equation (ODE)

$$\frac{dx(t)}{dt} = v(x(t), t) \qquad \text{with} \qquad x \in \mathbb{R}^d \,, \tag{5}$$

and a velocity field $v(x(t), t) \in \mathbb{R}^d$. This time evolution can be related to the underlying density through the continuity equation

$$\frac{\partial p(x,t)}{\partial t} + \nabla_x \left[p(x,t) v(x,t) \right] = 0 \,. \tag{6}$$

The velocity field transforms the density $p(x, t)$ such that

$$p(x,t) \to \begin{cases} \mathcal{N}(x; 0, 1) & t \to 0 \\ p_{\text{data}}(x) & t \to 1 \,. \end{cases} \tag{7}$$

We can estimate the velocity field with $v_\phi(x(t), t)$. In this case we can sample the data distribution from Gaussian random numbers, tracing the trajectory using any ODE solver. Defining the training trajectories to be linear, the velocity network is optimized using a simple MSE loss

$$\mathcal{L}_{\text{CFM}} = \left\langle \left[v_\phi((1-t)\epsilon + tx, t) - (x - \epsilon) \right]^2 \right\rangle_{t\sim U(0,1),\, \epsilon\sim\mathcal{N},\, x\sim p_{\text{data}}} \,. \tag{8}$$

Conditional probability distributions can be learned by allowing v_ϕ to depend on additional inputs.

For the energy network, we exploit the causal nature of the energy deposition in layers using an autoregressive transfusion architecture [87], as visualized in Fig 1. We start by embedding E_{inc} as our one-dimensional condition and the u-vector. For the u, this is done by concatenating a one-hot encoded position vector and zero-padding. These embeddings are passed to the encoder and decoder of a transformer, respectively. For the one-dimensional condition the encoder's self-attention reduces to a trivial 1×1 matrix. For the decoder we

[*]The code used for this paper is publicly available at https://github.com/heidelberg-hepml/calo_dreamer

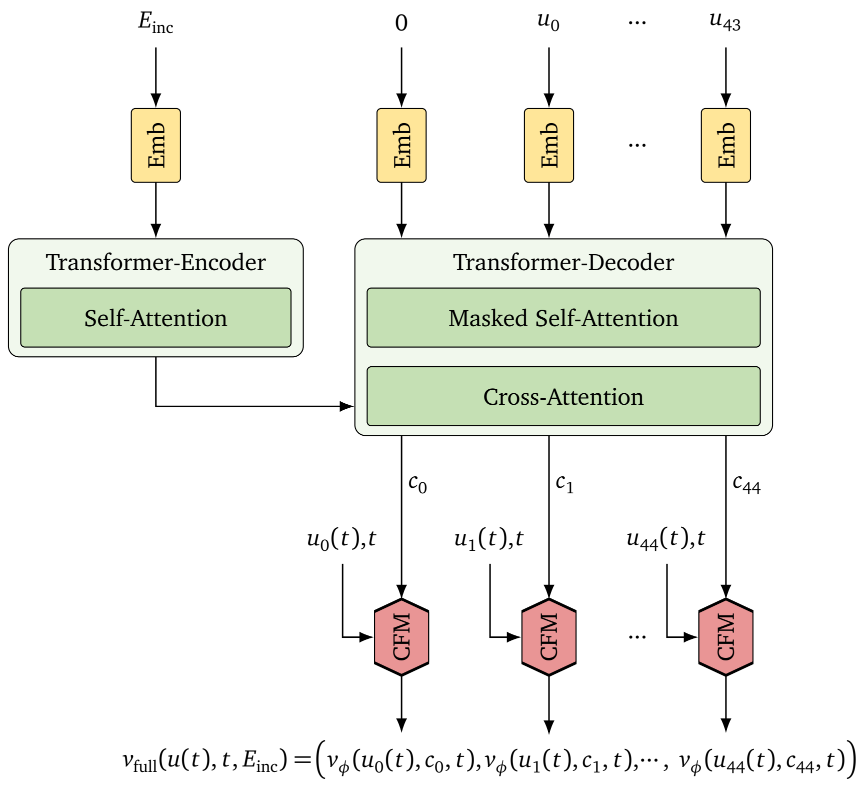


Figure 1: Schematic diagram of the autoregressive Transfusion network [87] used in our energy network.

mask our self-attention with an upper triangle matrix, to keep the autoregressive conditioning. Afterward, we apply a cross-attention between the encoder and decoder outputs. The transformer outputs the vectors $c_0, \ldots, c_{44}$, encoding the incident energy and previous energy ratios,

$$c_i = \begin{cases} c_i(u_0, \ldots, u_{i-1}, E_{\text{inc}}) & i > 0 \\ c_i(E_{\text{inc}}) & i = 0 \,. \end{cases} \tag{9}$$

For generation, we use a single dense CFM network v_ϕ, with the inputs time t, embedding c_i, and the point on the diffusion trajectory $u_i(t)$. This network is evaluated 45 times to predict each component of the velocity field individually,

$$v_{\text{full}}(u(t), t, E_{\text{inc}}) = \left(v_\phi(u_0(t), c_0, t), \ldots, v_\phi(u_{44}(t), c_{44}, t)\right) \tag{10}$$

During training, we can evaluate the contribution of each u_i to the loss in parallel, whereas sampling requires us to iteratively predict the u_i layer by layer. The hyperparameters of the transfusion network are given in Tab. 1.

3.2 Shape network — Vision Transformer

For the shape network, we use a 3-dimensional vision transformer (ViT) to learn the conditional velocity field $v_\theta(x(t), t, E_{\text{inc}}, u)$. The architecture is inspired by Ref [98] and illustrated in Fig. 2. It divides the calorimeter into non-overlapping groups of voxels, so-called patches, which are embedded using a shared linear layer and passed to a sequence of transformer

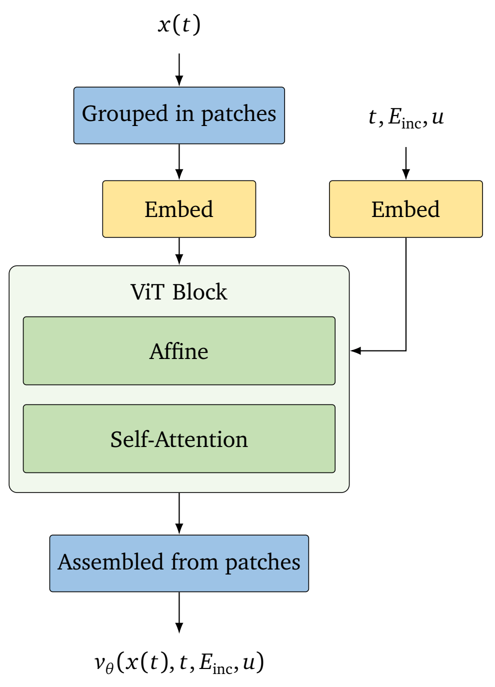


Figure 2: Schematic diagram of the vision transformer (ViT) [98] used in our shape network.

blocks. Each block consists of a multi-headed self-attention and a dense network that transforms the patch features. To break the permutation symmetry among patches, we add a learnable position encoding to the patch embeddings prior to the first attention block. After the last block, a linear layer projects the processed patch features into the original patch dimensions, where each entry represents a diffusion velocity. Finally, the patches are reassembled into the calorimeter shape.

The network uses a joint embedding for the conditional inputs, t, E_{inc} and u. The time and energy coordinates are embedded with separate dense networks, then summed into a single condition vector. The attention blocks incorporate this condition via affine transformations with shift and scale $a, b \in \mathbb{R}$ and an additional rescaling factor $\gamma \in \mathbb{R}$ learned by dense layers. These are applied within each block, and also to the final projection layer. Concretely, the operation inside the ViT block is summarized by

$$\begin{aligned} x_{\text{h}} &= x + \gamma_{\text{h}} g_{\text{h}}(a_{\text{h}} x + b_{\text{h}}), \\ x_{\text{l}} &= x_{\text{h}} + \gamma_{\text{l}} g_{\text{l}}(a_{\text{l}} x_{\text{h}} + b_{\text{l}}), \end{aligned} \tag{11}$$

where g_{h} is the multi-head self-attention step and g_{l} is the fully connected transformation. The hyperparameters of our transformer are given in Tab. 2.

The scalability of this architecture is closely tied to the choice of patching. On the one hand, small patches result in high-dimensional attention matrices. While this gives a more expressive network, the large number of operations can become a limitation for highly-granular calorimeters. Conversely, a large patch size compresses many voxels into one object, implying a faster forward pass but at the expense of sample quality. In this case, an expanded embedding dimension is needed to keep the network flexibility fixed. We decide on specific patch sizes for DS2 and DS3 through manual exploration.

Usually, we train Bayesian versions [99] of all our generative networks, including calorimeter showers [92]. In this study, the networks learning DS2 and DS3 are so heavy in terms of operations, that an increase by a factor two, to learn an uncertainty map over phase space, surpasses our typical training cost of 40 hours on a cutting-edge NVIDIA H100 GPU. In principle, Bayesian versions of all networks used in this study can be built and used to quantify limitations, for instance related to a lack of training data.

3.3 Latent diffusion

As the calorimeter granularity is increased from DS2 to DS3, the computational requirements to train a network on the full voxel space also increase considerably due to the larger number of patches. This motivates a study of how the naive scaling may be avoided by a lower-dimensional latent representation. Starting from the detector geometry, a voxel-based representation of a shower defines a grid with fixed size and stores the deposited energy in each voxel. This means a highly granular voxelization will produce a large fraction of zero voxels, but the showers should define a lower-dimensional manifold of the original phase space. Such a manifold can then be learned by an autoencoder [50, 92, 100].

We train a variational autoencoder with learnable parameters ψ. The encoder outputs a latent parameter pair (μ, σ), which defines the latent variable $r = \mu + z \cdot \sigma$ with $z \sim \mathcal{N}(0,1)$. The encoder distribution represents the phase space distributions over x through $p_\psi(r|x,u)$. For simplicity, in the following we drop the energy dependence in the encoder and decoder distributions. After sampling the latent variable, we minimize the learned likelihood of a Bernoulli decoder $p_\psi(x|r)$ represented by the reconstruction loss

$$\mathcal{L}_\text{VAE} = \Big\langle -\log p_\psi(x|r) \Big\rangle_{x \sim p_\text{data}, r \sim p_\psi(r|x)} + \beta \Big\langle D_\text{KL}[p_\psi(r|x), \mathcal{N}(0,1)] \Big\rangle_{x \sim p_\text{data}} . \qquad (12)$$

This choice of likelihood is possible since our preprocessing normalizes voxels into the range $[0,1]$. The reconstruction quality achieved in the autoencoder training places an upper bound on the quality of a generative model trained in the corresponding latent space.

The KL-divergence term, with unit-Gaussian prior and a small weight $\beta = 10^{-6}$, is a regularization rather than a condition for a tractable latent space. It encourages a smooth latent

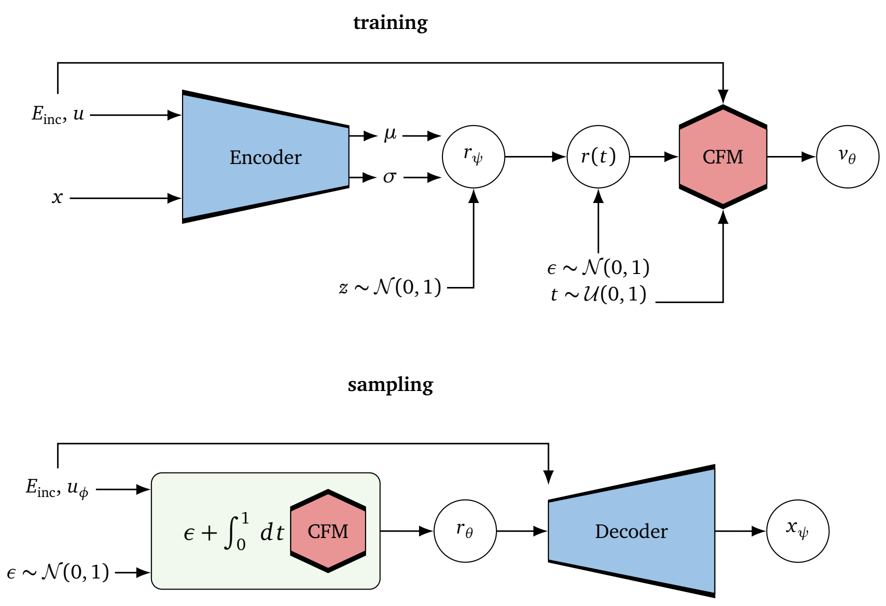


Figure 3: Training (upper) and sampling (lower) with the latent diffusion network, using a variational autoencoder.

space, over which we train the generative network. Especially for DS3, an autoencoder trained without KL-regularization produces a sparse latent space with features mapped over several orders of magnitude.

The VAE consists of a series of convolutions, the last of which downsamples the data. This structure is mirrored in the decoder using ConvTranspose operations. As always, the energy conditions are encoded in a separate network and passed to the encoder and decoder. For a compressed latent space the ratio between the dimensionality of x and r defines the reduction factor F. Rather than estimating the dimensionality of the datasets, we use a moderate, fixed reduction factor $F \simeq 2.5$ and a bottleneck with two channels. We do not expect the same reduction factor F to be optimal for both datasets. We provide more details on the autoencoder training in App. A.2.

The trained autoencoder is used as a pre- and postprocessing step for the CFM as illustrated in Fig. 3. Given the trained encoder distribution $p_\psi(r|x)$ the velocity field $v(r(t),t)$ imposes the boundary conditions

$$p(r,t) \to \begin{cases} \mathcal{N}(r;0,1) & t \to 0 \\ p_\psi(r|x) & t \to 1,\ x \sim p_{\text{data}}\,. \end{cases} \tag{13}$$

The expensive sampling then uses the lower-dimensional latent space and yields samples r from the learned manifold. Finally, the phase space configurations are provided by the deterministic decoder $D_\psi(r)$. Here we summarize the sampling procedure, including the energy dependence, as three sequential steps:

$$\begin{aligned} u &\sim p_\phi(u|E_{\text{inc}}) \\ r &\sim p_\theta(r,1|u,E_{\text{inc}}) \\ x &= D_\psi(r,u,E_{\text{inc}}) \end{aligned} \tag{14}$$

All network hyperparameters and the main training parameters are given in App. A.1.

3.4 Bespoke samplers

A potential drawback of CFM networks is their slower sampling than, for instance, normalizing flows with coupling layers [92]. This stems from the numerical integration of the ODE in Eq.(5). Depending on the complexity of the target distribution, a standard ODE solver requires $\mathcal{O}(100)$ steps to achieve high-fidelity samples, each consisting of at least one forward pass of the neural network.

One method to overcome this slow inference is distillation [57,63,101,102], which aims to predict the sampling trajectory at only a handful of intermediate points, or even at the terminus in a single step. This requires fine-tuning the network weights using additional training time, in some cases even additional training data. Further, since the weights of the network itself are updated, consistency is not strictly guaranteed and we can end up sampling from a different distribution than was originally learned.

An alternative approach is to keep the network fixed and consider alternative structures for the ODE solver. Reference [103] provides a comparison of various training-free solvers in the context of calorimeter simulations. While training-free approaches are the least costly, they are not task-specific and therefore unlikely to be optimal. However, there exists a trainable family of ODE solvers that can be optimized to a given vector field v_θ without excessive additional training [104, 105]. Such Bespoke Non-Stationary (BNS) solvers parameterize the steps along the flow trajectory. Starting from an initial state x_0, and a time discretization

$0 = t_0 < t_i < t_N = 1$, the i^{th} integration step is

$$x_{i+1} = a_i x_0 + b_i \cdot V_i \quad \text{with} \quad a_i \in \mathbb{R},\; b_i \in \mathbb{R}^{i+1}$$
$$V_i = [v_\theta(x_0, t_0), \cdots, v_\theta(x_i, t_i)] \in \mathbb{R}^{(i+1)\times d}\,, \tag{15}$$

where we again suppress the energy dependence of v_θ. By appropriately caching the velocities, each step requires just one evaluation of the network. In total, an N-step BNS solver has $N(N+5)/2+1$ learnable parameters: a_i, b_i and the t_i not fixed by the boundary conditions. Since this is typically orders of magnitude fewer than the network v_θ, optimizing the solver requires a fraction of the computation time needed to train the vector field itself. Non-stationary solvers encompass a large family of ODE solvers, including the Runge-Kutta (RK) methods. Euler's method, i.e. first order RK, corresponds to taking $a_i = 1, b_{ij} = 1/N$ and evenly-spaced t_i.

Bespoke solvers can be trained by comparing the bespoke trajectory to a precisely-computed reference $x_{\text{ref}}(t)$, given an initial state x_0 sampled from the CFM latent distribution. Here we define two options. First, the global truncation error measures the deviation between the final states of the solvers

$$\mathcal{L}_{\text{GTE}} = \Big\langle [x_{\text{ref}}(1) - x_N]^2 \Big\rangle_{x_0 \sim \mathcal{N}}\,, \tag{16}$$

where x_N is computed by iterating Eq.(15) starting from x_0. The local truncation error instead measures the discrepancy at each step,

$$\mathcal{L}_{\text{LTE}} = \Big\langle \sum_{i=0}^{N-1} \big[x_{\text{ref}}(t_{i+1}) - (a_i x_0 + b_i \cdot V_{\text{ref},i})\big]^2 \Big\rangle_{x_0 \sim \mathcal{N}}\,, \tag{17}$$

where $V_{\text{ref},i}$ is defined as in Eq.(15), but with velocities evaluated on the reference trajectory.

Although we use CFM for both our shape and energy networks, we only study BNS solvers for the shape network. For training a BNS solver, we initialize it to the Euler method. At each iteration, we sample an x_0 batch from the unit Gaussian and a batch of conditions from the energy network. A precise solver is then used to generate the reference trajectory $x_{\text{ref}}(t)$ which enters the loss. Note that the shape model parameters θ are frozen during training. The complete list of hyperparameters are given in App. A.1.

4 Results

4.1 Layer energies

In Fig. 4 we compare samples generated from the energy network with the truth for a selection of normalized layer energies u_i. The transfusion network indeed generates high-quality distributions, with errors comparable to the statistical uncertainties in the test data. The distributions for $u_{i>40}$ are the most difficult to model, since the majority of showers lie in the sharp peaks at zero or one. These are zero-width peaks corresponding to showers that end at the given layer, leading to a one, or end before or skip the layer, leading to a zero.

We find that our autoregressive setup is particularly effective in faithfully mapping regions close to these peaks. As a quantitative performance measure, we train a classifier to distinguish the u's defined by our energy network from the GEANT4 truth, obtaining AUC scores around 0.51 on an independent test set. The comparison in terms of layer energy is shown in Fig. 5. The factorization procedure allows us to use the same energy network for the ViT and the laViT, effectively generating statistically-identical layer energy distributions.

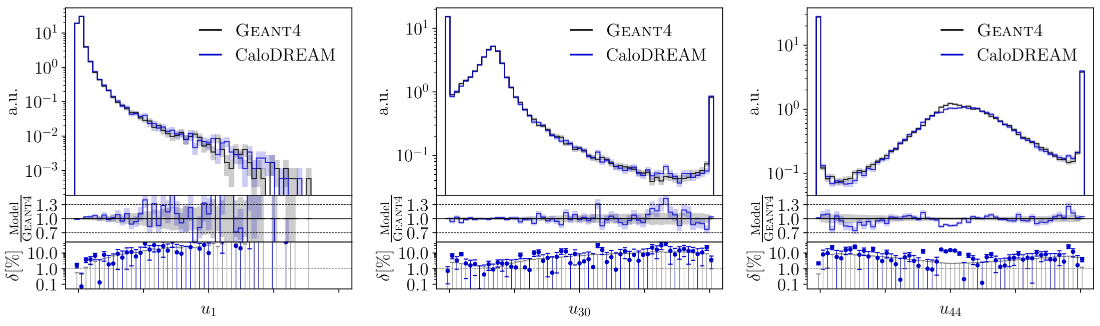


Figure 4: Distributions of selected u-features in DS2 from the CaloDREAM energy network (blue) compared to truth (grey). The error bars in all feature distributions in this paper show the statistics of the respective datasets.

4.2 DS2 showers

Given the learned layer energies, we use the shape networks described in Sec. 3.2 to generate the actual calorimeter showers over the voxels. First, we evaluate the distribution of energy depositions per layer by looking at shape observables, like the center of energy of the shower and its width in the ϕ and η directions,

$$\begin{aligned} \langle \xi \rangle &= \frac{\xi \cdot x}{\sum_i x_i} \\ \sigma_{\langle \xi \rangle} &= \sqrt{\frac{\xi^2 \cdot x}{\sum_i x_i} - \langle \xi \rangle^2} \qquad \text{for} \qquad \xi \in \{\eta, \phi\} \, . \end{aligned} \tag{18}$$

Here x_i is the energy deposition in a single voxel and the sum runs over the voxels in a layer.

In the first row of Fig. 5 we compare a set of layer-wise distributions from the networks trained in the full space and in the latent representation to the test data truth. We start with the energy deposited in layer 20, where for $E_{20} > 10$ MeV the full-dimensional vision transformer (ViT) as well as the latent-diffusion counterpart (laViT) agree with the truth at the level of a few per-cent, as expected. Towards smaller energies we see a missing feature in both networks. Also in the two other shown distributions the ViT and laViT agree with each other and deviate from GEANT4 only in regions with statistically limited training data.

The second row of Fig. 5 shows example distributions probing the combination of layers. In addition to the layer-wise shower shapes, we calculate the mean shower depth weighted by the energy deposition in each of the N layers for slices in the radial direction,

$$d_{r_j} = \frac{\sum_i^N k_i E_{i,r_j}}{E_{\text{tot},j}} \qquad r_j \in \{0, \dots, |r|\} \, . \tag{19}$$

Here E_{i,r_j} is the average energy deposition in slice r_j, and $E_{\text{tot},j}$ is the total energy deposition in the selected slice. Slices in the angular direction are less interesting to calculate due to the rotational invariance of the showers. This observable highlights a small deviation for both networks from the reference for showers with maximum depth of five layers not captured by the layer-wise high-level features.

Also combining layer-wise information, we show the total energy deposited in the calorimeter E_{tot} normalized by the incident energy and the full voxel distribution across the entire calorimeter E_{voxel}. The total shower energy relative to the incident energy is reproduced very well by both networks since this information is coming from the energy network. However for

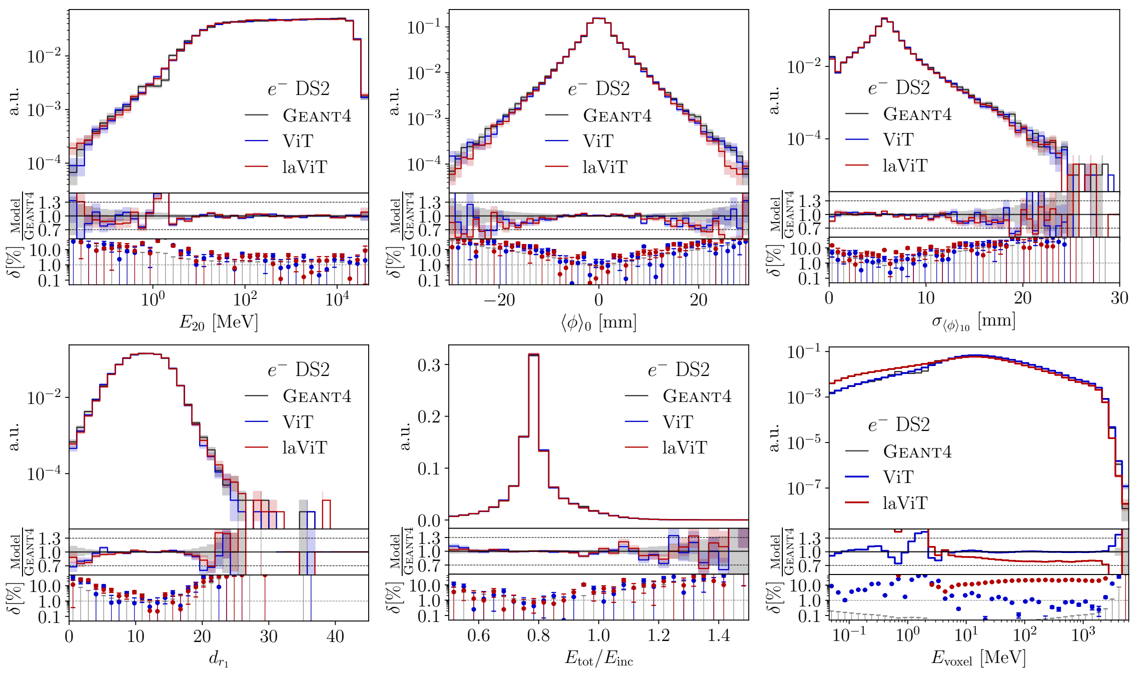


Figure 5: Selection of high-level features for DS2. The first row shows features for individual layers, the second row the combination of layers.

the voxel energies only the full-dimensional network captures the low-energy regime, whereas the latent model overestimates this regime and in turn shifts down the prediction for larger energies because of the normalization of the curve. This is the only noteworthy shortcoming of the laViT compared to the ViT that we find.

Following up on the problem raised by the last panel in Fig. 5, we focus on the (latent) description with low-energy voxels. In Fig. 6 we again compare the two network predictions with the truth, but applying an additional threshold cut of

$$E_{\text{voxel}} > 1 \text{ MeV} . \tag{20}$$

After this cut, the agreement of the laViT prediction with the full ViT and the truth improves significantly. We checked that this cut has only a limited impact on the total energy deposition E_{tot}. Slight deviations are limited to the threshold region $E_{\text{voxel}} \lesssim 5$ GeV. The reason can be seen in the sparsity distributions for instance of layer 10, λ_{10}. The laViT network generates a sizable number of showers with energy depositions everywhere, leading to a peak at zero

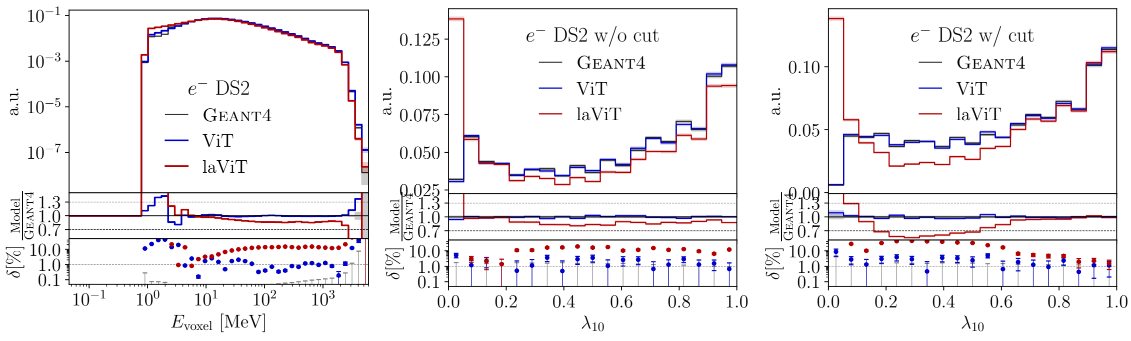


Figure 6: Effect of an additional threshold $E > 1$ MeV on DS2; we show the shower energy and the sparsities without and with threshold cut.

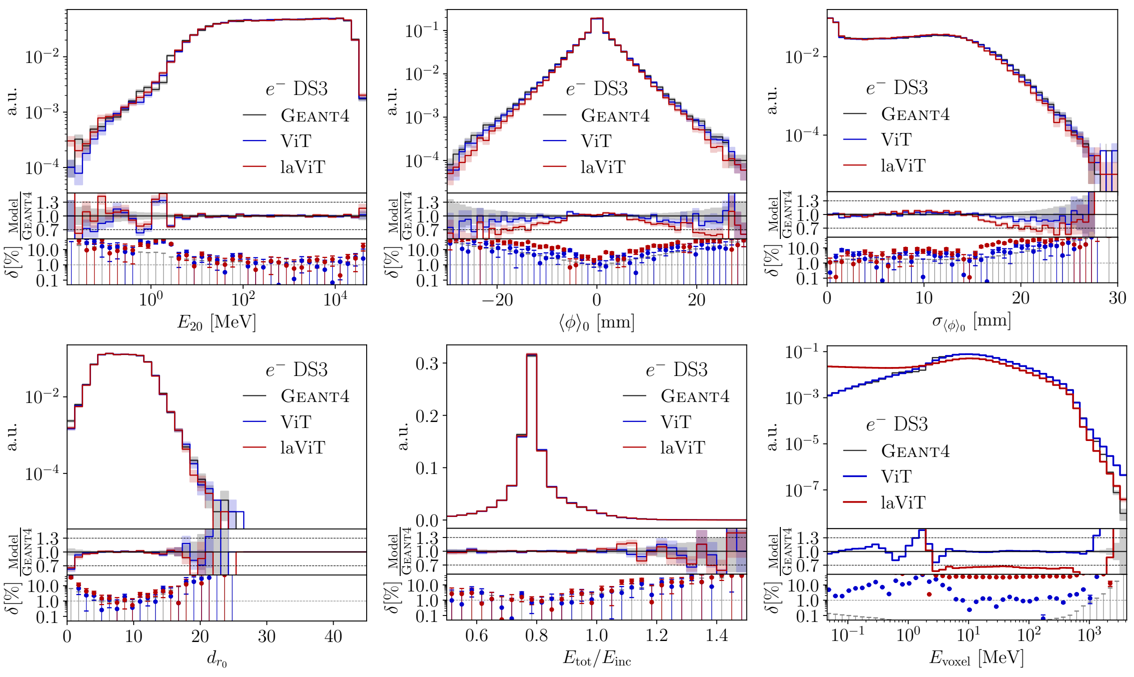


Figure 7: Selection of high-level features for DS3. The first row shows features for individual layers, the second row the combination of layers. All features correspond to the DS2 results shown in Fig. 5.

sparsity. This failure mode is already present in the autoencoder reconstruction as described in App. A.2. Because of their low energy, these contributions do not affect the other high-level observables or the learned physics patterns of the showers.

4.3 DS3 showers

The same analysis done for DS2 in Sec. 4.3 we now repeat for DS3. This means we study the same shower energies and shower shapes, but from 40500 instead of 6480 voxels. A target phase space of such large dimension is atypical for most LHC applications, and the key question is whether the precision-generative architectures that have been successful on lower-dimensional phase spaces also give the necessary precision for high-dimensional phase spaces. As a matter of fact, we know that this is not the case for standard normalizing flows or INNs [92], where the architectures have to be modified significantly to cope with higher resolution.

In Fig. 7 we again show a set of layer-wise features in the first row, observing extremely mild differences to the DS2 results. Only the shower shapes from the laViT suffer slightly in regions with too little training data. For the multi-layer features in the second row, we also find the same results as for DS2, including the challenge in describing voxels with $E_{\text{voxel}} \lesssim 3$ GeV.

Understanding and targeting this challenge, we again show the voxel energy distribution and the sparsity after the threshold cut $E_{\text{voxel}} > 1$ MeV in Fig. 8. For DS3 it turns out that after applying this cut the description of DS3 through the laViT network is excellent. The reason for this is two-fold. Given the low energy bound we can reproduce with the latent model, a cut larger than this threshold completely adjusts the sparsity up to a specific value by removing the additional energy deposition of the latent model and the noisy components of GEANT4. For both DS2 and DS3 the cut fixes the sparsity in λ_{10} down to $\lambda_{10} \gtrsim 0.7$. However, for DS3 this is done by moving the peak at zero, while for DS2 the mass is moved from the intermediate

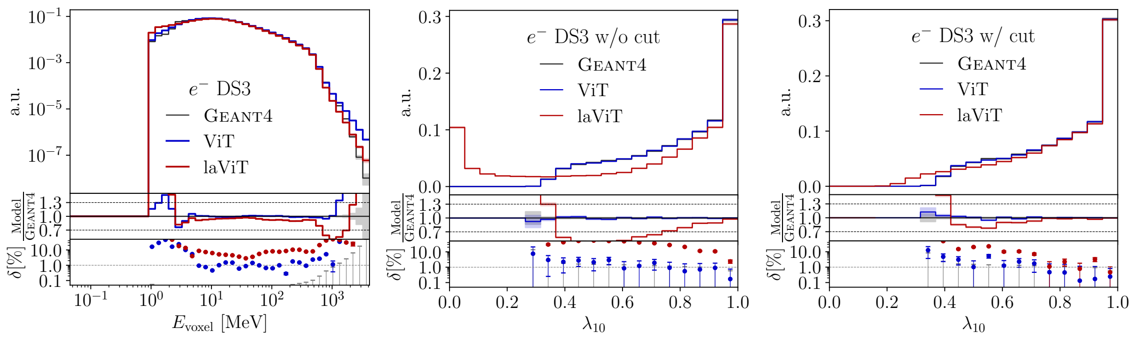


Figure 8: Effect of an additional threshold $E > 1$ MeV on DS3; we show the shower energy and the sparsities without and with threshold cut. All features correspond to the DS2 results shown in Fig. 6.

sparsity. This second difference comes from the dimensionalities of the two datasets, where the fixed reduction factor has a stronger impact on DS2 due to the larger information loss in the bottleneck.

4.4 Sampling efficiency

To demonstrate the performance of bespoke samplers, we compare the quality of showers produced by various solvers in terms of classifier tests. Classifiers trained to distinguish generated and true samples are an effective diagnostic tool since they capture failure modes in high-order correlations that are hidden in simple high-level distributions. As we will see in the following section, the phase space distribution of classifier scores can be used to search for and identify such failure modes. In this section, we only use the AUC as a simple, one-dimensional quality measure. The high-level classifier uses the layer-wise features but since we want sensitivity also to voxel-level correlations, we train a classifier on the low-level phase space as defined by the original voxels.

For our comparison, we include three standard fixed-steps solvers: the Euler, Midpoint, and Runge-Kutta 4 methods. We also consider bespoke non-stationary solvers using either the global, Eq.(16), or local, Eq.(17) truncation error as described in Sec. 3.4. Using each solver, we generate 100k showers from the DS2 ViT shape network. We train classifiers to distinguish these samples from the GEANT4 reference set using the standard CaloChallenge pipeline. In Fig. 9, we plot the high-level (left) and low-level (right) AUC scores against the number of function evaluations n_{eval} for each solver. Note that the Midpoint and RK4 methods respectively use 2 and 4 function evaluations per integration step. See App. A.3 for function evaluation timings of each network.

In both panels of the figure, we see that the Euler solver has a notably poor efficiency in terms of function evaluations. This indicates that the velocity field learned by the shape network has non-trivial curvature. Considering the remaining solvers, the sample quality essentially saturates by $n_{\text{eval}} = 64$ and all non-Euler methods appear to have statistically-equal performance at this point. The bespoke samplers demonstrate the best retention in quality when looking toward smaller n_{eval}. In particular, the local BNS solver keeps an AUC below 0.6 for both classifiers even at 8 function evaluations. The global BNS solver achieves a large margin of improvement at $n_{\text{eval}} = 4$ for the high-level classifier. The local bespoke solver also shows an advantage in the high-quality regime. Specifically, its AUC is already saturated for both solvers at 32 function evaluations. As such, in a resource-limited scenario the efficiency gains offered by bespoke solvers can be translated into improved sample quality.

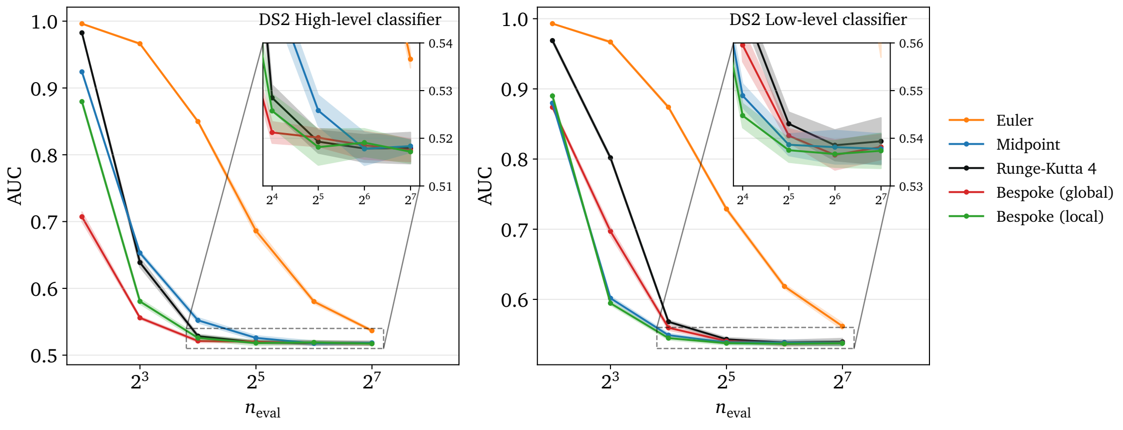


Figure 9: High-level (left) and low-level (right) classifier AUC scores on DS2 as a function of the number of function evaluations n_{eval} for various ODE solvers. Samples of 100k reference and generated showers are used to train the classifier. Errors bands are taken as the standard deviation over 10 runs.

It is interesting to note that the performance of a given solver can be significantly different between high- and low-level classifiers. This is evident in the reversed rankings of, for example, the two bespoke solvers in each panel. The global BNS solver favors performance on the high-level classifier, while the local BNS solver is best on the low-level classifier. A similar exchange can be seen among the Midpoint and RK4 solvers, with the former being close to optimal at low level. This distinction is only apparent away from the optimal number of function evaluations. For instance, at $n_{\text{eval}} = 8$ we observe a high-level classifier AUC score of ~ 0.55 for the global BNS and ~ 0.6 for the local BNS. The ranking, at the same point, is indeed reversed for the low-level classifier with an AUC of 0.7 for the global BNS and 0.6 for the local one.

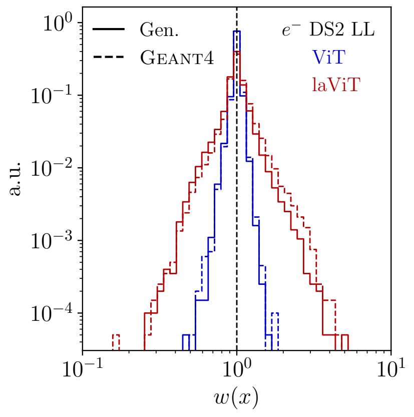
4.5 Performance

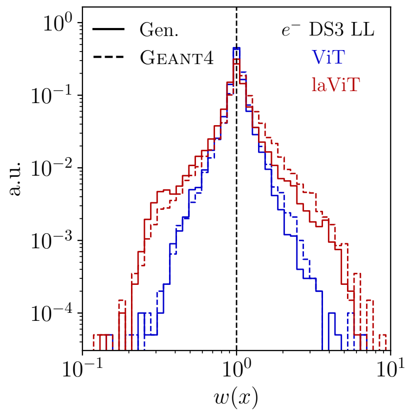
It is not trivial to test the overall performance of generative networks for calorimeter showers. In the previous sections we evaluated the networks using simple one-dimensional histograms, as in Figs. 5 and 7, or classifier AUC scores. A systematic approach to assess the quality of our generative networks, and a way to identify failure modes, is to examine the distribution of classifier predictions over the phase space or feature space x [91]. A properly trained and calibrated classifier $C(x)$ learns the likelihood-ratio between the data and the generated distributions which, according to the Neyman-Pearson lemma, is the most powerful test statistic to discriminate between the two samples. This allows us to extract a correction weight over phase space

$$w(x) = \frac{C(x)}{1 - C(x)} \approx \frac{p_{\text{data}}}{p_{\text{model}}}(x)\,, \tag{21}$$

and to use the corresponding weight distributions as an evaluation metric. The weights have to be evaluated on the training data and on the generated data, because failure modes appear as tails in one of the two distributions [91]. For example, if we only look at the weights of generated samples, we may not identify cases where the generator suffers from mode collapse. To further analyze failure cases, we can study showers with small or large weights as a function of phase space, using the interpretable nature of phase spaces in particle physics.

In Fig. 10 we show the classifier weights from the low-level classifier for DS2 and for DS3. We explicitly distinguish the weights for generated samples (solid lines) and GEANT4





	AUC (LL/HL)
	DS2
ViT	0.535(3)/0.517(3)
laViT	0.572(3)/0.536(2)
	DS3
ViT	0.63(1)/0.525(3)
laViT	0.62(1)/0.596(4)

Figure 10: Learned low-level (LL) classifier weight distributions for DS2 (left) and DS3 (right). We compare the full-dimensional ViT and the latent laViT results and, for each of them, show weights for the generated sample and for a GEANT4 test sample. The table shows the AUC values for both high-level (HL) and low-level classifiers in each case.

samples (dotted lines) obtained from the trained classifier. We also include a table with the AUC scores of the high-level classifier trained on layer-wise features and the low-level classifier, where the ViT shows state-of-the-art results on DS2 and the high-level DS3. The peaks of the weight distributions are nicely centered around $w = 1$, symmetric towards small and large (logarithmic) classifiers, and show no significant difference between generated and training data. The weights for the networks encoding the full phase space and the latent diffusion are different, with a typical broadening of the distribution by a factor two around the peak and larger and less smooth tails. We still observe that the classifier misses the low-energetic noise affecting the sparsity and the voxel energy distributions. Despite the simple nature of the neural network, a sequence of fully connected layers, the main result from this performance test is that the classifier identifies additional failure modes related to the step from DS2 to DS3 and to the reduced latent space. We expect these failure modes correspond to cross-layer features, since we observe a correlation between the classifier weights and the shower depth introduced in Sec. 4.2, and the high-level AUC is similar across the two datasets. Details of the neural network classifier are listed in App. A.1.

5 Outlook

Calorimeter showers are one of the most exciting applications of modern generative networks in fundamental physics. Their specific challenge is the high dimensionality of the voxelized phase space, combined with extremely sparse data and an LHC-level precision requirement. In our case, the CaloChallenge datasets 2 and 3 include up to 40k dimensions for the target phase space.

In this situation, diffusion networks allow us to go a step beyond standard normalizing flows. Our CaloDREAM architecture first factorizes the generation of detector showers into an energy network and a shape network. Both networks are trained using Conditional Flow Matching. The former generates the layer energies using a transformer backbone with self-attention and cross-attention blocks. For the latter, we use a 3-dimensional vision transformer, operating on patches of the target phase space.

For DS2 this combination of networks is a safe architecture choice, in the sense that it can be trained without problems and reproduces all features, within layers and across layers, with high precision. We can use a VAE to reduce the dimensionality using latent diffusion. We find essentially no loss in performance, except for the reproduction of low-energy voxels and, with it, sparsity, which can be improved by introducing an MeV-level energy threshold. Because diffusion networks are slower than alternative generative networks, we use bespoke samplers to enhance their generation speed, at no cost of the precision and improving the fidelity in case of limited resources.

For DS3 the performance of the CaloDREAM generators remains qualitatively the same, but the shape network reaches the limit in terms of available computation time. This is a motivation to again employ latent diffusion. We find excellent performance of the latent diffusion architecture; with the right choice of energy thresholds even the sparsity distribution is reproduced correctly. However, irrespective of the dataset, further studies are needed to understand the effects of mapping the distributions into real detectors with irregular geometries, more complex distributions from different incident particles, e.g. hadrons, and varying angle of impact.

Our study shows that modern generative networks can be used to describe calorimeter showers in highly granular calorimeters. When the number of phase space dimensions becomes very large and the data becomes sparse, a latent diffusion network combined with an (autoregressive) transformer and bespoke sampling provides excellent benchmarks in speed and in precision. We publish the generated samples together with the full set of high-level features in the Zenodo repository 10.5281/zenodo.14413046.

Note added A potentially similar approach, CaloDiT, also using a diffusion transformer to tackle calorimeter showers has been shown at ACAT 2024.

Acknowledgements

We would like to thank Theo Heimel, Lorenz Vogel, and Anja Butter for extremely helpful discussions. This research is supported through the KISS consortium (05D2022) funded by the German Federal Ministry of Education and Research BMBF in the ErUM-Data action plan, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant 396021762 – TRR 257: *Particle Physics Phenomenology after the Higgs Discovery*, and through Germany's Excellence Strategy EXC 2181/1 – 390900948 (the *Heidelberg STRUCTURES Excellence Cluster*). SPS is supported by the BMBF Junior Group Generative Precision Networks for Particle Physics (DLR 01IS22079). The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG (bwForCluster NEMO)

A Further details

A.1 Hyperparameters

Parameter	DS2 & DS3
Epochs	500
LR sched.	cosine
Max LR	10^{-3}
Batch size	4096
ODE solver	Runge-Kutta 4 (50 steps)
Network	transformer
Dim embedding	64
Intermediate dim	1024
Num heads	4
Num layers	4
Network	dense feed-forward
Intermediate dim	256
Num layers	8
Activation	SiLU

Table 1: Parameters for the autoregressive energy network in Sec. 3.1.

	ViT		laViT	
Parameter	DS2	DS3	DS2	DS3
Patch size	(3, 16, 1)	(3, 5, 2)	(3, 1, 1)	(3, 2, 2)
Embedding dimension	480	240	240	240
Attention heads	6	6	6	6
MLP hidden dimension	1920	720	960	960
Blocks	6	6	10	10
epochs	800	600	800	400
batch size	64	64	128	128
LR sched.	cosine			
Max LR	10^{-3}			
ODE solver	Runge-Kutta 4 (20 steps)			

Table 2: Parameters for the shape networks in Sec. 3.2, for the full and the latent space.

Parameter	Value
Optimizer	Adam
Learning rate	$2 \cdot 10^{-4}$
Batch size	1000
Epochs	200
Number of layers	3
Hidden nodes	512
Dropout	20%
Activation function	leaky ReLU
Training samples	70k
Validation samples	10k
Testing samples	20k

Table 3: Parameters for HL and LL classifiers network used to calculate the weights of Fig. 10. The other classifiers use the same hyperparameters but without any dropout.

Parameter	Value	
	DS2	DS3
Loss	BCE + βKL	
β	10^{-6}	
Epochs	200	
Out activation	sigmoid	
Lr sched.	OneCycle	
Max lr	10^{-3}	
# of blocks	2 (+ bottleneck)	
Channels	(64, 64, 2)	
Dim. bottleneck	(2, 15, 9, 9)	(2, 9, 26, 16)
Kernels	[(3,2,1), (1,1,1)]	[(5,2,3), (1,1,1)]
Strides	[(3,2,1), (1,1,1)]	[(2,2,1), (1,1,1)]
Paddings	[(0,1,0), (0,0,0)]	[(0,1,0), (0,0,0)]
Normalized cut	$1 \cdot 10^{-6}$	

Table 4: Parameters of the autoencoder for DS2 and DS3 used for the laViT network in Sec. 3.3.

Parameter	Value
Reference solver	midpoint (100 steps)
Initialization	Euler
Optimizer	Adam
Learning rate	$1 \cdot 10^{-3}$
Batch size	100
Max iterations	5000
Stopping patience (iterations)	200

Table 5: Parameters used to train BNS solvers, described in Sec. 3.4.

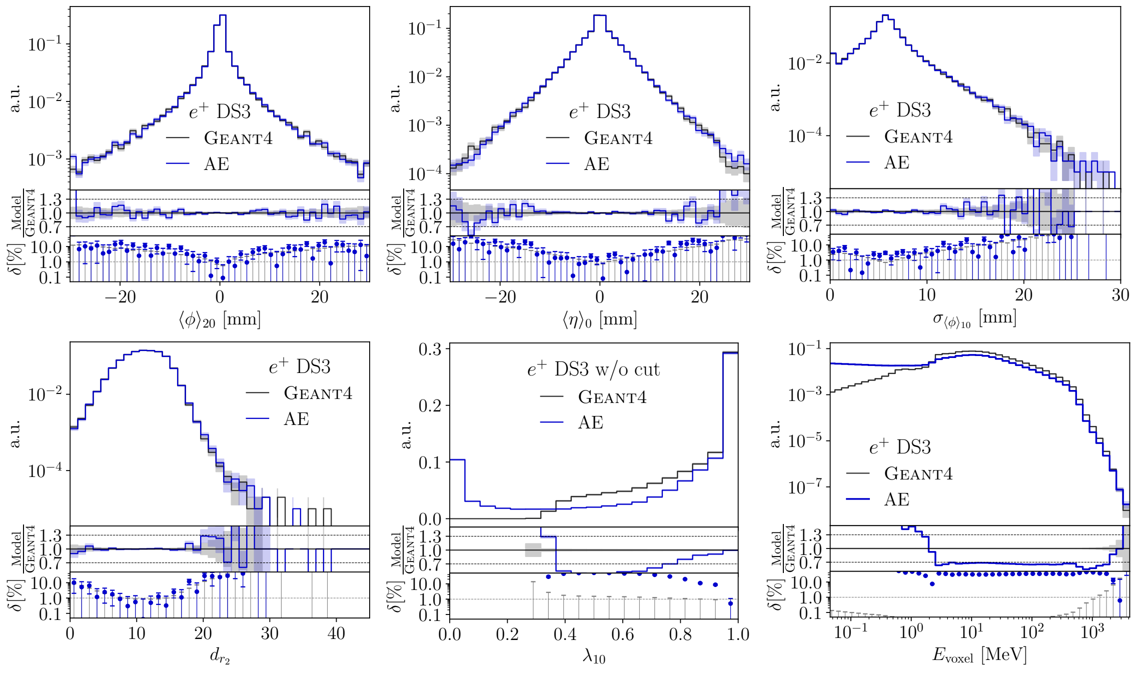


Figure 11: Selection of high-level features sensitive to the reconstruction of the autoencoder for DS3.

A.2 Autoencoder

The VAE introduced in Sec. 3.3 is trained separately, using the BCE reconstruction loss

$$\mathcal{L}_{\text{VAE}} = -\Big\langle x\log(x_\psi) + (1-x)\log(1-x_\psi)\Big\rangle_{p_\psi(r|x)} + \beta D_{\text{KL}}[p_\psi(r|x), \mathcal{N}(0,1)]\,. \tag{22}$$

This loss provides notably better reconstruction quality than the standard MSE loss, both in terms of high-level features and a neural network classifier trained to distinguish reconstructed showers from an independent test set. A detailed description of the network architecture is provided in Tab. 4. Each block consists of three Conv2d operations that preserve the number of channels of which the final one downsamples according to the stride and padding parameters. In addition, we break the translation equivariance by adding the coordinates of each input to the activation map as new channels [58].

In Fig. 11 we provide a set of kinematic distributions similar to Fig. 7 for DS3, to illustrate the VAE reconstruction. We find that the only missing feature in the learned manifold is the distribution of the low-energetic voxels, also reflected in the sparsity. We also train a classifier using the hyperparameters of Tab. 3 on the low-level features which gives an AUC score of 0.512(5) consistently for both, DS2 and DS3.

A.3 Timing

In Sec. 4.4, we study the sampling cost of networks in terms of the number of function evaluations n_{eval}. Here we provide timing measurements for a single forward pass of each of our CFM networks, using a batch size 100. We ran tests using a single NVIDIA H100 GPU and summarize the results in Tab. 6. The times for the energy network are identical across the two datasets since there is no change in the network architecture. Also note that since the energy model is autoregressive, sampling with an N-step solver uses $N \times L$ function evaluations, where L is the number of calorimeter layers. As we did not perform an extensive hyperparameter search, we expect there to be room for improvements for all listed models.

Network	Time (ms) on GPU	
	DS2	DS3
Energy	0.37±0.01	0.37±0.01
Shape (ViT)	17±1	84±8
Shape (LaViT)	31±1	63±6

Table 6: Timings for one network forward pass using batch size 100 on an NVIDIA H100.

References

[1] J. M. Campbell *et al.*, *Event generators for high-energy physics experiments*, SciPost Phys. **16** (2024) 5, 130, arXiv:2203.11110 [hep-ph].

[2] S. Badger *et al.*, *Machine learning and LHC event generation*, SciPost Phys. **14** (2023) 4, 079, arXiv:2203.07460 [hep-ph].

[3] GEANT4, S. Agostinelli *et al.*, *GEANT4 - A Simulation Toolkit*, Nucl. Instrum. Meth. A **506** (2003) 250.

[4] J. Allison *et al.*, *Geant4 developments and applications*, IEEE Transactions on Nuclear Science **53** (2006) 1, 270.

[5] J. Allison *et al.*, *Recent developments in geant4*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **835** (2016) 186.

[6] HEP Software Foundation, J. Apostolakis *et al.*, *HEP Software Foundation Community White Paper Working Group - Detector Simulation*, arXiv:1803.04165 [physics.comp-ph].

[7] ATLAS, *ATLAS Software and Computing HL-LHC Roadmap*, tech. rep., CERN, Geneva, 2022.

[8] T. Plehn, A. Butter, B. Dillon, T. Heimel, C. Krause, and R. Winterhalder, *Modern Machine Learning for LHC Physicists*, arXiv:2211.01421 [hep-ph].

[9] J. Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks*, arXiv:1707.00028 [hep-ph].

[10] M. D. Klimek and M. Perelstein, *Neural Network-Based Approach to Phase Space Integration*, SciPost Phys. **9** (2020) 053, arXiv:1810.11509 [hep-ph].

[11] I.-K. Chen, M. D. Klimek, and M. Perelstein, *Improved neural network Monte Carlo simulation*, SciPost Phys. **10** (2021) 1, 023, arXiv:2009.07819 [hep-ph].

[12] C. Gao, J. Isaacson, and C. Krause, *i-flow: High-dimensional Integration and Sampling with Normalizing Flows*, Mach. Learn. Sci. Tech. **1** (2020) 4, 045023, arXiv:2001.05486 [physics.comp-ph].

[13] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, and S. Schumann, *Exploring phase space with Neural Importance Sampling*, SciPost Phys. **8** (2020) 4, 069, arXiv:2001.05478 [hep-ph].

[14] C. Gao, S. Höche, J. Isaacson, C. Krause, and H. Schulz, *Event Generation with Normalizing Flows*, Phys. Rev. D **101** (2020) 7, 076002, arXiv:2001.10028 [hep-ph].

[15] K. Danziger, T. Janßen, S. Schumann, and F. Siegert, *Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates*, SciPost Phys. **12** (2022) 164, arXiv:2109.11964 [hep-ph].

[16] T. Heimel, R. Winterhalder, A. Butter, J. Isaacson, C. Krause, F. Maltoni, O. Mattelaer, and T. Plehn, *MadNIS - Neural multi-channel importance sampling*, SciPost Phys. **15** (2023) 4, 141, arXiv:2212.06172 [hep-ph].

[17] T. Janßen, D. Maître, S. Schumann, F. Siegert, and H. Truong, *Unweighting multijet event generation using factorisation-aware neural networks*, SciPost Phys. **15** (2023) 3, 107, arXiv:2301.13562 [hep-ph].

[18] E. Bothmann, T. Childers, W. Giele, F. Herren, S. Hoeche, J. Isaacson, M. Knobbe, and R. Wang, *Efficient phase-space generation for hadron collider event simulation*, SciPost Phys. **15** (2023) 4, 169, arXiv:2302.10449 [hep-ph].

[19] T. Heimel, N. Huetsch, F. Maltoni, O. Mattelaer, T. Plehn, and R. Winterhalder, *The MadNIS reloaded*, SciPost Phys. **17** (2024) 1, 023, arXiv:2311.01548 [hep-ph].

[20] L. de Oliveira, M. Paganini, and B. Nachman, *Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis*, Comput. Softw. Big Sci. **1** (2017) 1, 4, arXiv:1701.05927 [stat.ML].

[21] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, *JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics*, Eur. Phys. J. C **79** (2019) 2, 102, arXiv:1804.09720 [hep-ph].

[22] E. Bothmann and L. Debbio, *Reweighting a parton shower using a neural network: the final-state case*, JHEP **01** (2019) 033, arXiv:1808.07802 [hep-ph].

[23] K. Dohi, *Variational Autoencoders for Jet Simulation*, arXiv:2009.04842 [hep-ph].

[24] E. Buhmann, G. Kasieczka, and J. Thaler, *EPiC-GAN: Equivariant point cloud generation for particle jets*, SciPost Phys. **15** (2023) 4, 130, arXiv:2301.08128 [hep-ph].

[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*, SciPost Phys. **16** (2024) 1, 018, arXiv:2303.05376 [hep-ph].

[26] V. Mikuni, B. Nachman, and M. Pettee, *Fast point cloud generation with diffusion models in high energy physics*, Phys. Rev. D **108** (2023) 3, 036025, arXiv:2304.01266 [hep-ph].

[27] 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*, arXiv:2310.00049 [hep-ph].

[28] P. Ilten, T. Menzo, A. Youssef, and J. Zupan, *Modeling hadronization using machine learning*, SciPost Phys. **14** (2023) 3, 027, arXiv:2203.04983 [hep-ph].

[29] A. Ghosh, X. Ju, B. Nachman, and A. Siodmok, *Towards a deep learning model for hadronization*, Phys. Rev. D **106** (2022) 9, 096020, arXiv:2203.12660 [hep-ph].

[30] J. Chan, X. Ju, A. Kania, B. Nachman, V. Sangli, and A. Siodmok, *Fitting a deep generative hadronization model*, JHEP **09** (2023) 084, arXiv:2305.17169 [hep-ph].

[31] C. Bierlich, P. Ilten, T. Menzo, S. Mrenna, M. Szewc, M. K. Wilkinson, A. Youssef, and J. Zupan, *Towards a data-driven model of hadronization using normalizing flows*, SciPost Phys. **17** (2024) 2, 045, arXiv:2311.09296 [hep-ph].

[32] M. Paganini, L. de Oliveira, and B. Nachman, *Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters*, Phys. Rev. Lett. **120** (2018) 4, 042003, arXiv:1705.02355 [hep-ex].

[33] L. de Oliveira, M. Paganini, and B. Nachman, *Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters*, J. Phys. Conf. Ser. **1085** (2018) 4, 042017, arXiv:1711.08813 [hep-ex].

[34] 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) 1, 014021, arXiv:1712.10321 [hep-ex].

[35] M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt, *Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks*, Comput. Softw. Big Sci. **2** (2018) 1, 4, arXiv:1802.03325 [astro-ph.IM].

[36] M. Erdmann, J. Glombitza, and T. Quast, *Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network*, Comput. Softw. Big Sci. **3** (2019) 1, 4, arXiv:1807.01954 [physics.ins-det].

[37] D. Belayneh *et al.*, *Calorimetry with deep learning: particle simulation and reconstruction for collider physics*, Eur. Phys. J. C **80** (2020) 7, 688, arXiv:1912.06794 [physics.ins-det].

[38] 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) 1, 13, arXiv:2005.05334 [physics.ins-det].

[39] ATLAS Collaboration, *Fast simulation of the ATLAS calorimeter system with Generative Adversarial Networks*, Tech. Rep. ATL-SOFT-PUB-2020-006, CERN, Geneva, Nov, 2020.

[40] 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 [physics.ins-det].

[41] C. Krause and D. Shih, *Fast and accurate simulations of calorimeter showers with normalizing flows*, Phys. Rev. D **107** (2023) 11, 113003, arXiv:2106.05285 [physics.ins-det].

[42] ATLAS, G. Aad *et al.*, *AtlFast3: The Next Generation of Fast Simulation in ATLAS*, Comput. Softw. Big Sci. **6** (2022) 1, 7, arXiv:2109.02551 [hep-ex].

[43] C. Krause and D. Shih, *Accelerating accurate simulations of calorimeter showers with normalizing flows and probability density distillation*, Phys. Rev. D **107** (2023) 11, 113004, arXiv:2110.11377 [physics.ins-det].

[44] 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) 2, 025014, arXiv:2112.09709 [physics.ins-det].

[45] C. Chen, O. Cerri, T. Q. Nguyen, J. R. Vlimant, and M. Pierini, *Analysis-Specific Fast Simulation at the LHC with Deep Learning*, Comput. Softw. Big Sci. **5** (2021) 1, 15.

[46] 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 [hep-ph].

[47] V. Mikuni and B. Nachman, *Score-based generative models for calorimeter shower simulation*, Phys. Rev. D **106** (2022) 9, 092009, arXiv:2206.11898 [hep-ph].

[48] ATLAS, G. Aad *et al.*, *Deep Generative Models for Fast Photon Shower Simulation in ATLAS*, Comput. Softw. Big Sci. **8** (2024) 1, 7, arXiv:2210.06204 [hep-ex].

[49] C. Krause, I. Pang, and D. Shih, *CaloFlow for CaloChallenge dataset 1*, SciPost Phys. **16** (2024) 5, 126, arXiv:2210.14245 [physics.ins-det].

[50] J. C. Cresswell, B. L. Ross, G. Loaiza-Ganem, H. Reyes-Gonzalez, M. Letizia, and A. L. Caterini, *CaloMan: Fast generation of calorimeter showers with density estimation on learned manifolds*, in *36th Conference on Neural Information Processing Systems: Workshop on Machine Learning and the Physical Sciences*. 11, 2022. arXiv:2211.15380 [hep-ph].

[51] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh, and D. Shih, *L2LFlows: generating high-fidelity 3D calorimeter images*, JINST **18** (2023) 10, P10017, arXiv:2302.11594 [physics.ins-det].

[52] B. Hashemi, N. Hartmann, S. Sharifzadeh, J. Kahn, and T. Kuhr, *Ultra-high-granularity detector simulation with intra-event aware generative adversarial network and self-supervised relational reasoning*, Nature Commun. **15** (2024) 1, 4916, arXiv:2303.08046 [physics.ins-det]. [Erratum: Nature Commun. 115, 5825 (2024)].

[53] A. Xu, S. Han, X. Ju, and H. Wang, *Generative machine learning for detector response modeling with a conditional normalizing flow*, JINST **19** (2024) 02, P02003, arXiv:2303.10148 [hep-ex].

[54] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, K. Krüger, P. McKeown, and L. Rustige, *New angles on fast calorimeter shower simulation*, Mach. Learn. Sci. Tech. **4** (2023) 3, 035044, arXiv:2303.18150 [physics.ins-det].

[55] 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*, JINST **18** (2023) 11, P11025, arXiv:2305.04847 [physics.ins-det].

[56] M. R. Buckley, C. Krause, I. Pang, and D. Shih, *Inductive simulation of calorimeter showers with normalizing flows*, Phys. Rev. D **109** (2024) 3, 033006, arXiv:2305.11934 [physics.ins-det].

[57] V. Mikuni and B. Nachman, *CaloScore v2: single-shot calorimeter shower simulation with diffusion models*, JINST **19** (2024) 02, P02001, arXiv:2308.03847 [hep-ph].

[58] O. Amram and K. Pedro, *Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation*, Phys. Rev. D **108** (2023) 7, 072014, arXiv:2308.03876 [physics.ins-det].

[59] S. Diefenbacher, V. Mikuni, and B. Nachman, *Refining Fast Calorimeter Simulations with a Schrödinger Bridge*, arXiv:2308.12339 [physics.ins-det].

[60] M. Faucci Giannelli and R. Zhang, *CaloShowerGAN, a generative adversarial network model for fast calorimeter shower simulation*, Eur. Phys. J. Plus **139** (2024) 7, 597, arXiv:2309.06515 [physics.ins-det].

[61] I. Pang, D. Shih, and J. A. Raine, *Calorimeter shower superresolution*, Phys. Rev. D **109** (2024) 9, 092009, arXiv:2308.11700 [physics.ins-det].

[62] M. Leigh, D. Sengupta, J. A. Raine, G. Quétant, and T. Golling, *Faster diffusion model with improved quality for particle cloud generation*, Phys. Rev. D **109** (2024) 1, 012010, arXiv:2307.06836 [hep-ex].

[63] 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*, JINST **19** (2024) 04, P04020, arXiv:2309.05704 [physics.ins-det].

[64] J. Birk, E. Buhmann, C. Ewen, G. Kasieczka, and D. Shih, *Flow Matching Beyond Kinematics: Generating Jets with Particle-ID and Trajectory Displacement Information*, arXiv:2312.00123 [hep-ph].

[65] S. Schnake, D. Krücker, and K. Borras, *CaloPointFlow II Generating Calorimeter Showers as Point Clouds*, arXiv:2403.15782 [physics.ins-det].

[66] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri, and R. Verheyen, *Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer*, Nature Commun. **12** (2021) 1, 2985, arXiv:1901.00875 [hep-ph].

[67] B. Hashemi, N. Amin, K. Datta, D. Olivito, and M. Pierini, *LHC analysis-specific datasets with Generative Adversarial Networks*, arXiv:1901.05282 [hep-ex].

[68] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghighat, and S. Palazzo, *DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC*, JHEP **08** (2019) 110, arXiv:1903.02433 [hep-ex].

[69] A. Butter, T. Plehn, and R. Winterhalder, *How to GAN LHC Events*, SciPost Phys. **7** (2019) 6, 075, arXiv:1907.03764 [hep-ph].

[70] Y. Alanazi *et al.*, *Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)*, arXiv:2001.11103 [hep-ph].

[71] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot, and S. Vent, *Generative networks for precision enthusiasts*, SciPost Phys. **14** (2023) 4, 078, arXiv:2110.13632 [hep-ph].

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

[73] A. Butter, T. Jezo, M. Klasen, M. Kuschick, S. Palacios Schweitzer, and T. Plehn, *Kicking it off(-shell) with direct diffusion*, SciPost Phys. Core **7** (2024) 3, 064, arXiv:2311.17175 [hep-ph].

[74] K. Datta, D. Kar, and D. Roy, *Unfolding with Generative Adversarial Networks*, arXiv:1806.00433 [physics.data-an].

[75] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, and R. Winterhalder, *How to GAN away Detector Effects*, SciPost Phys. **8** (2020) 4, 070, arXiv:1912.00477 [hep-ph].

[76] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, and J. Thaler, *OmniFold: A Method to Simultaneously Unfold All Observables*, Phys. Rev. Lett. **124** (2020) 18, 182001, arXiv:1911.09107 [hep-ph].

[77] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone, and U. Köthe, *Invertible Networks or Partons to Detector and Back Again*, SciPost Phys. **9** (2020) 074, arXiv:2006.06685 [hep-ph].

[78] M. Backes, A. Butter, M. Dunford, and B. Malaescu, *An unfolding method based on conditional invertible neural networks (cINN) using iterative training*, SciPost Phys. Core **7** (2024) 1, 007, arXiv:2212.08674 [hep-ph].

[79] M. Leigh, J. A. Raine, K. Zoch, and T. Golling, *ν-flows: Conditional neutrino regression*, SciPost Phys. **14** (2023) 6, 159, arXiv:2207.00664 [hep-ph].

[80] J. A. Raine, M. Leigh, K. Zoch, and T. Golling, *Fast and improved neutrino reconstruction in multineutrino final states with conditional normalizing flows*, Phys. Rev. D **109** (2024) 1, 012005, arXiv:2307.02405 [hep-ph].

[81] 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 [hep-ex].

[82] J. Ackerschott, R. K. Barman, D. Gonçalves, T. Heimel, and T. Plehn, *Returning CP-observables to the frames they belong*, SciPost Phys. **17** (2024) 1, 001, arXiv:2308.00027 [hep-ph].

[83] S. Diefenbacher, G.-H. Liu, V. Mikuni, B. Nachman, and W. Nie, *Improving generative model-based unfolding with Schrödinger bridges*, Phys. Rev. D **109** (2024) 7, 076011, arXiv:2308.12351 [hep-ph].

[84] N. Huetsch *et al.*, *The Landscape of Unfolding with Machine Learning*, arXiv:2404.18807 [hep-ph].

[85] S. Bieringer, A. Butter, T. Heimel, S. Höche, U. Köthe, T. Plehn, and S. T. Radev, *Measuring QCD Splittings with Invertible Networks*, SciPost Phys. **10** (2021) 6, 126, arXiv:2012.09873 [hep-ph].

[86] A. Butter, T. Heimel, T. Martini, S. Peitzsch, and T. Plehn, *Two invertible networks for the matrix element method*, SciPost Phys. **15** (2023) 3, 094, arXiv:2210.00019 [hep-ph].

[87] T. Heimel, N. Huetsch, R. Winterhalder, T. Plehn, and A. Butter, *Precision-Machine Learning for the Matrix Element Method*, arXiv:2310.07752 [hep-ph].

[88] 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) 11, P11004, arXiv:2009.03796 [hep-ph].

[89] R. Winterhalder, M. Bellagente, and B. Nachman, *Latent Space Refinement for Deep Generative Models*, arXiv:2106.00792 [stat.ML].

[90] B. Nachman and R. Winterhalder, *Elsa: enhanced latent spaces for improved collider simulations*, Eur. Phys. J. C **83** (2023) 9, 843, arXiv:2305.07696 [hep-ph].

[91] R. Das, L. Favaro, T. Heimel, C. Krause, T. Plehn, and D. Shih, *How to understand limitations of generative networks*, SciPost Phys. **16** (2024) 1, 031, arXiv:2305.16774 [hep-ph].

[92] F. Ernst, L. Favaro, C. Krause, T. Plehn, and D. Shih, *Normalizing Flows for High-Dimensional Detector Simulations*, arXiv:2312.09290 [hep-ph].

[93] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska, "Fast calorimeter simulation challenge 2022 - dataset 2." https://doi.org/10.5281/zenodo.6366271, March, 2022.

[94] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska, "Fast calorimeter simulation challenge 2022 - dataset 3." https://doi.org/10.5281/zenodo.6366324, March, 2022.

[95] O. Amram *et al.*, *CaloChallenge 2022: A Community Challenge for Fast Calorimeter Simulation*, arXiv:2410.21611 [cs.LG].

[96] M. Faucci Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska, "Fast calorimeter simulation challenge 2022 github page." https://github.com/CaloChallenge/homepage, 2022.

[97] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, *Flow matching for generative modeling*, arXiv preprint arXiv:2210.02747 (2022) .

[98] W. S. Peebles and S. Xie, *Scalable diffusion models with transformers*, 2023 IEEE/CVF International Conference on Computer Vision (ICCV) (2022) 4172.

[99] M. Bellagente, M. Haussmann, M. Luchmann, and T. Plehn, *Understanding Event-Generation Networks via Uncertainties*, SciPost Phys. **13** (2022) 1, 003, arXiv:2104.04543 [hep-ph].

[100] Q. Liu, C. Shimmin, X. Liu, E. Shlizerman, S. Li, and S.-C. Hsu, *Calo-VQ: Vector-Quantized Two-Stage Generative Model in Calorimeter Simulation*, arXiv:2405.06605 [physics.ins-det].

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

[102] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, *Consistency models*, in *Proceedings of the 40th International Conference on Machine Learning*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds. PMLR, 23–29 Jul, 2023.

[103] C. Jiang, S. Qian, and H. Qu, *Choose Your Diffusion: Efficient and flexible ways to accelerate the diffusion model in fast high energy physics simulation*, arXiv:2401.13162 [physics.ins-det].

[104] N. Shaul, J. C. Perez, R. T. Q. Chen, A. K. Thabet, A. Pumarola, and Y. Lipman, *Bespoke solvers for generative flow models*, ArXiv **abs/2310.19075** (2023) .

[105] N. Shaul, U. Singer, R. T. Q. Chen, M. Le, A. Thabet, A. Pumarola, and Y. Lipman, *Bespoke non-stationary solvers for fast sampling of diffusion and flow models*, ArXiv **abs/2403.01329** (2024) .