AllShowers: One model for all calorimeter showers

Thorsten Buss[1,2] ⋆, Henry Day-Hall[2], Frank Gaede[2], Gregor Kasieczka[1] and Katja Krüger[2]

1 Institut für Experimentalphysik, Universität Hamburg, Hamburg, Germany
2 Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

⋆ Thorsten.Buss@uni-hamburg.de

Abstract

Accurate and efficient detector simulation is essential for modern collider experiments. To reduce the high computational cost, various fast machine learning surrogate models have been proposed. Traditional surrogate models for calorimeter shower modeling train separate networks for each particle species, limiting scalability and reuse. We introduce AllShowers, a unified generative model that simulates calorimeter showers across multiple particle types using a single generative model. AllShowers is a continuous normalizing flow model with a Transformer architecture, enabling it to generate complex spatial and energy correlations in variable-length point cloud representations of showers. Trained on a diverse dataset of simulated showers in the highly granular ILD detector, the model demonstrates the ability to generate realistic showers for electrons, photons, and charged and neutral hadrons across a wide range of incident energies and angles without retraining. In addition to unifying shower generation for multiple particle types, AllShowers surpasses the fidelity of previous single-particle-type models for hadronic showers. Key innovations include the use of a layer embedding, allowing the model to learn all relevant calorimeter layer properties; a custom attention masking scheme to reduce computational demands and introduce a helpful inductive bias; and a shower- and layer-wise optimal transport mapping to improve training convergence and sample quality. AllShowers marks a significant step towards a universal model for calorimeter shower simulations in collider experiments.

Received Date
Accepted Date
Published Date

Contents

1 Introduction

Simulations of particle detectors in high-energy physics (HEP) experiments incur high computational costs, which are expected to increase beyond available resources in the near future [1, 2]. Fast generative models must substitute the most expensive MC simulation steps to achieve sufficient statistics with the available computing resources. To fully realize the physics potential of new experiments with higher event rates and highly granular calorimeters, more accurate and efficient fast generative models must be developed.

Many techniques for fast calorimeter simulation have been explored for existing or similar to existing experiments; generative adversarial networks (GANs) [3–16], variational autoencoders (VAEs) [14, 17–19], classical normalizing flows (NFs) [20–30], auto-regressive models [31], and diffusion and continuous flow models [32–40]. Even the highly granular pixel vertex detector of Belle II has been simulated using a GAN [41]. Future calorimeter designs have also been specifically targeted in various generative modeling projects including; GANs [42], VAEs [43–48], NFs [49, 50] and continuous flow models [47, 51–56].

These methods can be seen in comparison in a recent taxonomy of detector simulation [57], and for the simulation of current detector designs, the accuracy and efficiency of many variants was compared in the CaloChallenge 2022 [58].

The model put forward in this paper breaks new ground; to the best of our knowledge no previous model has captured the response of both the electromagnetic calorimeter (ECAL) and hadronic calorimeter (HCAL) with such a comprehensive set of particle species, let alone for a highly granular Higgs factory detector.[1] This combination of twelve particle types in a single model would be challenging at current granularities, but is even more challenging with the high granularity expected in future calorimeters.

[1] When preparing this manuscript for submission, [56] was released. It attempts a similar task, albeit for a smaller set of different particles.

A multi-particle model, such as this one, is needed for three reasons: firstly, in production environments, maintaining code infrastructure uses significant human and computational resources, and by handling twelve particles together we simplify the codebase, reducing the cost of both validation and maintenance. Secondly, fast calorimeter simulation occurs within full scale MC simulations, so the models must share local memory resources with many other components, which quickly becomes a limitation on model size and therefore performance. By combining particle types, common physics behavior will be shared between particles resulting in better use of local memory, and so facilitating more accurate modeling. Finally, it is hoped that fast calorimeter simulation might benefit a wider range of users than just the production environments at major experiments. These users could save energy and compute time on more dedicated small scale tasks that might require custom installations. A comprehensive model for all particle showers reduces the technical overhead of setup and installation, increasing the utilization and usefulness of the model for these users.

To achieve this, we introduce AllShowers, a continuous normalizing flow (CNF) model with a Transformer architecture. AllShowers is trained on a diverse dataset of simulated showers in the highly granular calorimeters of the International Large Detector (ILD) [59]. The model consists of two components: the PointCountFM, which predicts the number of points per layer conditioned on incident particle information, and the CNF-transformer, which generates the position and energy of each point additionally conditioned on the layer index of each point. AllShowers has several significant improvements compared to its predecessor models [47,54]. Using an embedding layer for the calorimeter layer index allows the model to learn all relevant calorimeter layer properties, such as material budget and distance from the calorimeter surface, from data. A custom attention masking scheme is employed to reduce computational demands and introduce a helpful inductive bias, allowing points to attend only to points in nearby layers. Additionally, a shower- and layer-wise optimal transport mapping is used to improve convergence during training and sample quality.

The layout of this paper is as follows. In the next section, section 2, the dataset is described. This includes a summary description of the detector chosen as an example of a detector system with high granularity calorimeters, the particle gun used for shower generation, and the data preprocessing. Following this, in section 3, the architecture of the AllShowers model is presented, along with a description of the training process. Then the results are presented in section 4. Finally, in section 5, the paper is concluded with a discussion of the findings.

2 Dataset

We used the International Large Detector (ILD) [59] as an example of a detector with highly granular sampling calorimeters. The ILD was initially designed for the International Linear Collider (ILC), a proposed electron-positron collider, and could be adapted for other future colliders. The ILD detector design is optimized for particle-flow algorithms, which reconstruct particles with high precision by combining information from multiple subdetectors.

The ILD calorimeter system consists of a highly granular electromagnetic calorimeter (ECAL) [60] and a hadronic calorimeter (HCAL). Both of which sit within a superconducting coil generating a magnetic field of 3.5 T strength. The ECAL is composed of 30 layers with tungsten absorbers and silicon sensors with about 5×5 mm^2 pads. For mechanical reasons and to reduce dead material, two active layers are always mounted on either side of a tungsten support. This results in a small modulation in measured energy in even and odd layers. To improve energy resolution at low energies while preserving good confinement of most EM showers, two different absorber thicknesses are used: a smaller one for the first 20 layers and a larger one for the last 10 layers. The HCAL consists of 48 layers with stainless steel absorbers

and polystyrene scintillator tiles measuring about 3×3 cm^2.

As in earlier work [54, 61], we use a regularized readout geometry without insensitive gaps between calorimeter modules. This broadens the model's applicability to other incident point locations. Hits, which the model produces in inactive material, will be dropped when integrated into the full simulation chain.

Using Geant4 [62] and the DD4hep [63] framework, we simulated a dataset of four million showers originating from twelve different incident particle types, namely: e^-, e^+, π^-, π^+, K^-, K^+, K^0_L, p, $\bar{p}$, n, $\bar{n}$, and γ. The incident particle type is randomly chosen for each shower with equal probability. We cover all incident angles of particles originating at the interaction point (IP) and reaching the calorimeter barrel region, including magnetic-field effects. This means that the angular bounds depend on the incident particles charge and energy. The energy of the incident particles is uniformly distributed between 5 GeV and 130 GeV. A random sample of 50k showers is used as a validation set. For testing, we simulated several datasets with the incident particle distributions and statistics given in the results section.

2.1 Data Representation

The calorimeter shower data are represented as a 4D point cloud of energy depositions (Geant4 steps) in active material, where each point is represented as a tuple (x, y, z, e). Here, x and y denote the local coordinates in millimeters, with x aligned along the direction of the magnetic field, and z indicates the layer index, ranging from 0 to 77 (covering both ECAL and HCAL layers). e represents the deposited energy.

To reduce the number of points while preserving geometry independence, the energy depositions are binned into a grid that is three times finer in the two transversal dimensions, i.e., nine times higher granularity, than the respective readout pads [51]. For each non-empty bin, a point is created using the x and y coordinates of the highest energy deposition within the bin, and the total energy within the bin.

We counteract the incident angle dependence of the shower shape by shifting the x and y coordinates of each point such that the incident particle always appears to enter the calorimeter perpendicularly at the origin. While this transformation does not eliminate all angle dependencies, it significantly simplifies the model's learning task. To further reduce the number of points, we remove all points with an energy deposit below 10 keV or with a time of over 200 ns (bunch crossing window). The time constructed is already applied before clustering. We place a quadratic bounding box around the shower core removing all points outside this box. The side length is chosen to be the side length of the octagon formed by the ECAL surface (c.a. 1500 mm). This will exclude most of the points for which the flat layer assumption breaks. The excluded energy depositions are far away from the shower core and typically low in energy. After these preprocessing steps, the average number of points per shower is 2306, with a maximum of 6006.

For preprocessing, the x and y coordinates are rescaled to have standard deviation one and mean zero (Standardization), z is kept as the discrete layer index, and the logarithm of the energy is also standardized. Points are zero-padded to a maximum of 6016 points per shower for batch training. 6016 is a multiple of 128, making the computation of attention masks easier and more efficient.

3 Model and Training

The AllShowers model consists of two main components: the PointCountFM and the CNF-transformer, as illustrated in figure 1. The PointCountFM is responsible for generating the number of points per layer conditioned on the incident particle information (particle type,

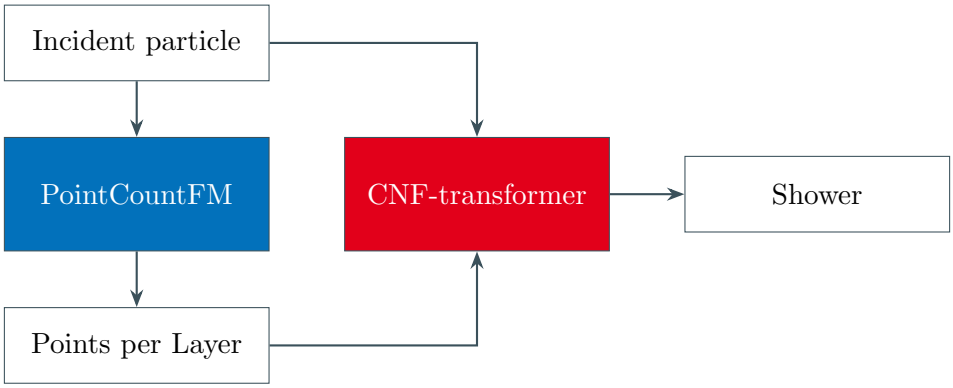

Figure 1: Schematic overview of the AllShowers model architecture. The PointCountFM predicts the number of points per layer, which are then used by the CNF-transformer to generate the full shower. The incident particle information is provided to both models. The point layer index, needed by the CNF-transformer, can be computed from the number of points per layer.

energy and angle). After initializing as many points as demanded by PointCountFM, the CNF-transformer generates the position within the layer and the energy of each point, again conditioned on the incident particle information. The layer index is provided as an additional condition. Splitting the model into two components in this way is inspired by CaloFlow [20] where the layer-wise energy depositions are generated first and has been used in various other works.

Both models use the continuous normalizing flow (CNF) [64] paradigm as a means to model the complex distributions of calorimeter showers. In CNFs, the transformation from latent to physics space is modeled as the solution of an ordinary differential equation (ODE) $\frac{d\mathbf{x}_t}{dt} = v_c(x_t, t)$, where $v_c(\mathbf{x}_t, t)$ is a neural network that predicts the vector field, t is the integration variable, and c is the condition. $\mathbf{x}_0$ is the initial condition, a sample from the latent space. $\mathbf{x}_1$ is a physics space sample. Note that x denotes the spacial coordinate in the calorimeter while $\mathbf{x}_t$ denotes an entire data sample. During generation a numerical ODE solver is used.

A likelihood based training of CNFs is possible [64], but computationally inefficient. Instead, we use the recently proposed conditional flow matching (FM) [65] approach. In FM, the vector field is constructed as the expectation value of all straight lines connecting physics and latent space samples. This mean squared error is evaluated using Monte Carlo integration over physics and latent space. FM has been shown to be more efficient than likelihood-based training for CNFs [65].

3.1 PointCountFM

The PointCountFM was already introduced in CaloHadronic [47]. It is responsible for generating the number of points per layer, 78 integers in total, conditioned on the incident particle information (type, energy, angle). The type is given as a one-hot encoded vector, the energy is converted to logarithmic scale and standard scaled, and the angle is represented by a vector on the unit sphere. This is a generalization of the approach taken in CaloHadronic, where only fixed angle and particle type were considered.

As an additional improvement, we no longer use dequantization noise during training. While dequantization is essential for classical likelihood-based training of flows on discrete data, it is not necessary for FM. We found that removing the dequantization noise leads to a significant improvement in performance especially for low point counts. Dequantization works well when a change by one in the discrete value has no significant effect on the downstream

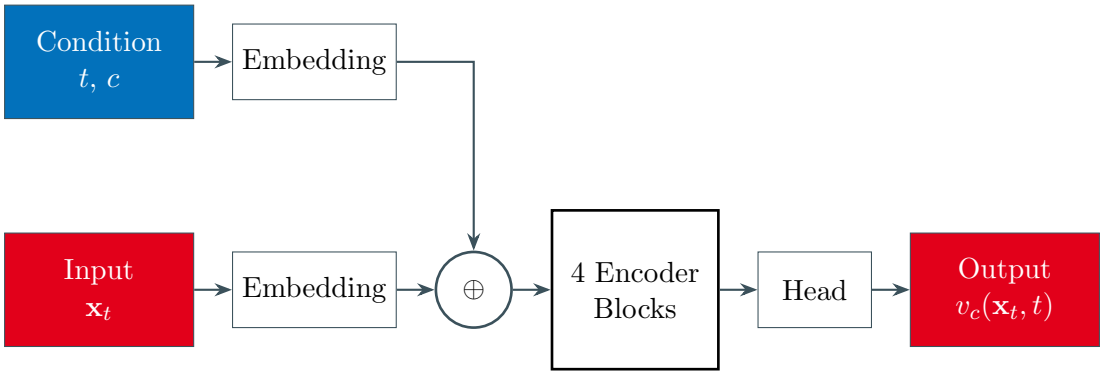

Figure 2: Schematic overview of the CNF-transformer architecture. The input $\mathbf{x}_t$ for $t = 0$ is a standard normal sample, for $t = 1$ it is the preprocessed shower. Since the calorimeter layer is a condition, $\mathbf{x}_t$ is a three-dimensional point-cloud (x_k, y_k, e_k). The condition c includes the incident particle information and the layer index. t is the time variable of the neural ODE. The output $v_c(\mathbf{x}_t, t)$ is the vector field used in the CNF, e.g. the right-hand side of the neural ODE.

task. However, in our case, a change from zero to one point in a layer can confuse the CNF-transformer significantly, as it has to generate a point in an unexpected layer.

A complete list of hyper-parameters can be found in Appendix C.

3.2 CNF-Transformer

After PointCountFM has predicted the number of points per layer (n_i), as many latent space points as requested are initialized. The first n_0 points are assigned to layer 0, the next n_1 points to layer 1, and so on. Each point is initialized with a standard normal sample in the x, y, and $\log(e)$ dimensions. The layer index, z, is provided as an additional condition. Then CNF-transformer transforms these points into a calorimeter shower.

An overview of the CNF-transformer architecture is shown in figure 2. The input is the point cloud, $\mathbf{x}_t$, at ODE time t. For $t = 0$, this is a standard normal sample, and for $t = 1$, it is the preprocessed shower. The output is the vector field $v_c(\mathbf{x}_t, t)$ used in the CNF, i.e. the right-hand side of the neural ODE. We can split the condition on global conditioning information and point-wise conditioning information. The global conditioning information includes the incident particle type, energy, and angle, while the point-wise conditioning information is the layer index. Input, time, and conditions are embedded and element-wise summed. The resulting representation is processed by four transformer encoder blocks. Finally, a head network produces the output vector field.

3.3 Embeddings

The main purpose of the embeddings is to map the different inputs to a common feature space.

Input Embedding The input $\mathbf{x}_t$ is a point cloud of shape $(N, 3)$, where N is the number of points. We embedded each point independently using a single linear layer going from 3 to 64 dimensions.

Time Embedding For the time embedding, we used the standard Fourier feature mapping [66] with 3 frequencies, followed by a linear layer going from 6 to 64 dimensions.

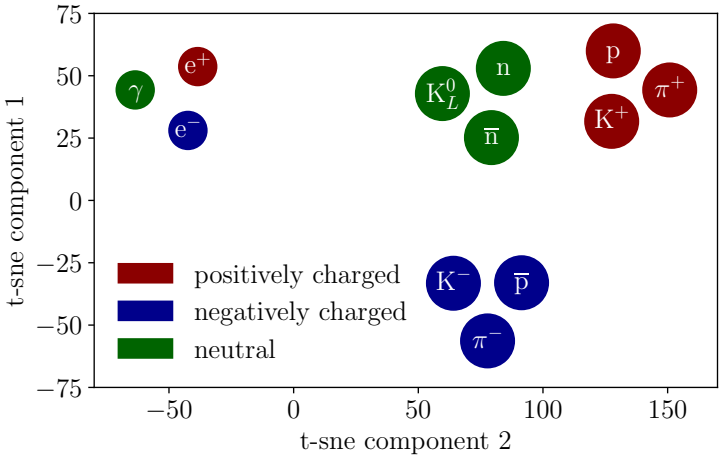

Figure 3: 2D t-SNE [67] visualization of the learned 64 dimension particle type embeddings. Small circles indicate electromagnetic showers, large circles indicate hadronic showers. One can see four distinct clusters: one for electromagnetic showers, one for positively charged hadrons, one for negatively charged hadrons, and one for neutral hadrons.

Condition Embedding The global conditional information includes the incident particle type, energy, and direction. The model explicitly learns 12 embedding vectors, one for each particle type. Figure 3 shows a t-SNE [67] visualization of these embeddings. The preprocessed energy and direction are concatenated and passed through a linear layer going from 4 to 64 dimensions. Since the point layer is provided as point-wise information, the model has an implicit conditioning on the number of points per layer. To make it explicit, we also provide the number of points per layer as global information. The number of points per layer is passed through a feedforward network with one hidden layer of size 128 with ReLU activation, going from 78 to 64 dimensions.

Layer Embedding The calorimeter layer index is provided as point-wise conditional input. The model explicitly learns 78 embedding vectors, one for each of the 78 layers. This allows the model to learn layer-specific features like distance from the ECAL surface, material budgets, and typical energy deposition.

After embedding, the global features are repeated for each point and all features are summed element-wise.

3.4 Fast Attention Masking

One major drawback of transformers is their quadratic complexity in the number of input tokens. In our dataset, the number of points per shower can be up to 6016, which would require more than 36 million attention weights. In the computer science literature, various methods for masking attention weights have been proposed. Most notably, the Sparse Transformer [69], Longformer [70], and BigBird [71] architectures. However, these methods have been developed in the context of natural language processing, where the input is a sequence. In our case, the input is a point cloud without any inherent ordering.

We developed a custom attention masking scheme that takes advantage of the fact that points are grouped by calorimeter layer. We allow points that are up to two layers apart to attend to each other. This means that points in layer i can attend to points in layers $i-2$, $i-1$,

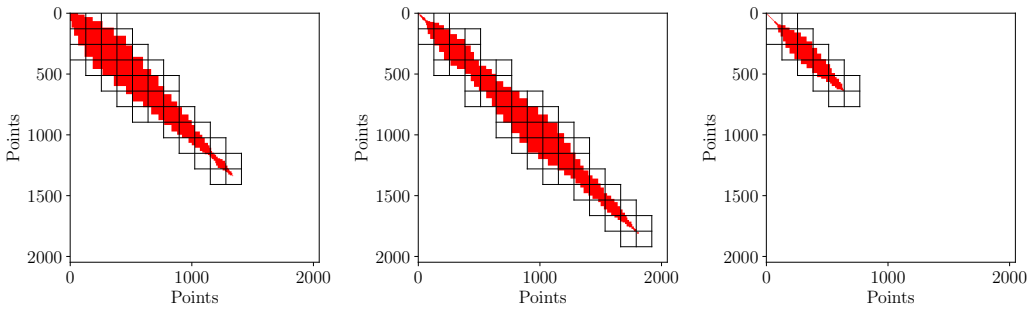

Figure 4: Examples of attention masks for three different showers. Shown is a part of the full 6016 × 6016 attention matrix where each entry indicates whether two points can attend to each other. Red entries indicate allowed attention, white entries indicate masked attention. The black lines indicate the 128 × 128 blocks flex-attention [68] will compute. White entries within these blocks are computed but then masked out.

i, $i+1$, and $i+2$. In combination with padding masking [72], this leads to a high degree of sparsity in the attention weights for our dataset. To utilize this sparsity, we used PyTorch's [73] built-in FlexAttention [68] module. For this to work efficiently, input points are sorted by layer index before starting the training. An example of attention masks for three different showers is shown in figure 4.

This attention masking scheme leads to a significant speed-up during training and inference; roughly a factor of twenty which is roughly considered with the sparsity of the attention weights. This allowed us to train the CNF-transformer for more epochs leading to better performance. We also found improved performance with the same number of training epochs, indicating that the inductive bias introduced by the attention masking is beneficial.

3.5 Layer-wise Optimal Transport Mapping

In continuous normalizing flows and diffusion models, the sampling process involves transforming samples from a simple latent distribution (e.g., a standard normal distribution) to match the complex data distribution. To do so, an ordinary or stochastic differential equation (ODE or SDE) is solved, which can be resource intensive. The number of function evaluations (NFE) necessary to get good results is strongly correlated with the curvature of the trajectories taken by samples during the transformation $\kappa = |\ddot{\mathbf{x}}_t|$, where $\ddot{\mathbf{x}}_t$ is the second derivative of the sample with respect to the integration variable t.

The main reason CNFs have curvature in their trajectories is the random mapping of data points to latent points during training. To overcome this problem, batch-wise optimal transport (OT) mapping has been proposed [74]. The idea is to approximate the optimal mapping between data points and latent points which would lead to straight trajectories. To achieve this, the optimal transport problem is solved for each batch during training. However, this approach is only feasible for generative problems without or with simple conditioning.

Instead of mapping data and latent points, we map physics point-cloud points to latent point-cloud points exploiting the permutation invariance. Since the calorimeter layer conditioning breaks permutation invariance, the OT mapping is only applied per shower and layer. We solve the OT problem using the Python Optimal Transport (POT) library [75]. The cost function is the Euclidean distance in the 3D space of preprocessed points.

The layer-wise OT mapping leads to shorter trajectories, faster training convergence, and better results.

3.6 Training Details

We trained the CNF-transformer using the Lookahead optimizer [76] with RAdam [77, 78] as the inner optimizer and decoupled weight decay [79]. We found this combination, also known as Ranger [80], to be especially robust against training instabilities, leading to reliable convergence in our experiments. We wrote a custom Ranger implementation in PyTorch to fit our needs. As learning rate scheduler, we used a cosine annealing schedule. Since RAdam has an integrated warm-up phase, we did not use an additional warm-up schedule. We trained the model with a batch size of 256 for 200 epochs. The training took less than 24 hours on 16 Nvidia A100 GPUs. A complete list of hyper-parameters can be found in Appendix C.

3.7 Energy Calibration

After training, we found that the total energy per shower generated by the CNF-transformer was too low by approximately 3.3% on average. While a simple rescaling of the point energies could fix this, it would influence other distributions in a negative way. Instead, we rescale the incident energy we provide as condition to the CNF-transformer by a factor of 1.033 during inference. This simple calibration step fixes the total energy per shower without negatively impacting other distributions.

4 Results

In the following section, the performance of the model is presented on multiple levels. Beginning at the single event scale, in section 4.1, the ability of the model to generate realistic detailed events is shown. Secondly, in section 4.2, the ensemble-level distributions of the model are compared to the targets they seek to replicate. Following this, in sections 4.3 and 4.4, the ensemble-level distributions are compared to other models with similar objectives, and finally, section 4.5 looks at the inference speed of this model.

In order to render all comparisons fair, the same post processing is applied to the output of all models. Hits produced by the models are clustered into regular grids intended to resemble the granularity of the calorimeter in question; so in the ECAL, hits have been clustered into cells of 5×5 millimeters, and in the HCAL, into cells of 30×30 millimeters. Each model has used its own conventions for training data preprocessing, and we do not wish the relationship between the grid in post processing and any grids imposed on the training data to introduce artifacts, therefore we add a random offset to the post processing grid in each event. Finally, cells with energy below half the energy deposited by a Minimum Ionizing Particle (MIP) are conventionally removed before reconstruction to reduce electronics noise, so we remove these cells in the post processing as well.

4.1 Individual Showers

One of the more exciting features of a high granularity calorimeter is how distinctly it resolves particle showers from different particle types. In figure 5, we can see examples of six particle types, each shown once as simulated by Geant4 (upper) and once by AllShowers (lower). The direction and energy chosen is the same for each model, and the number of points per layer is fixed to be that chosen by the Geant4 simulation, that is to say, for AllShowers, only the CNF-transformer is used, PointCountFM does not run. Thus the two models are compelled to generate events with similar depth for each shower, and the results are directly comparable. In each image, a gap can bee seen at about $z = 2015$ mm where the ECAL ends and the HCAL has yet to start.

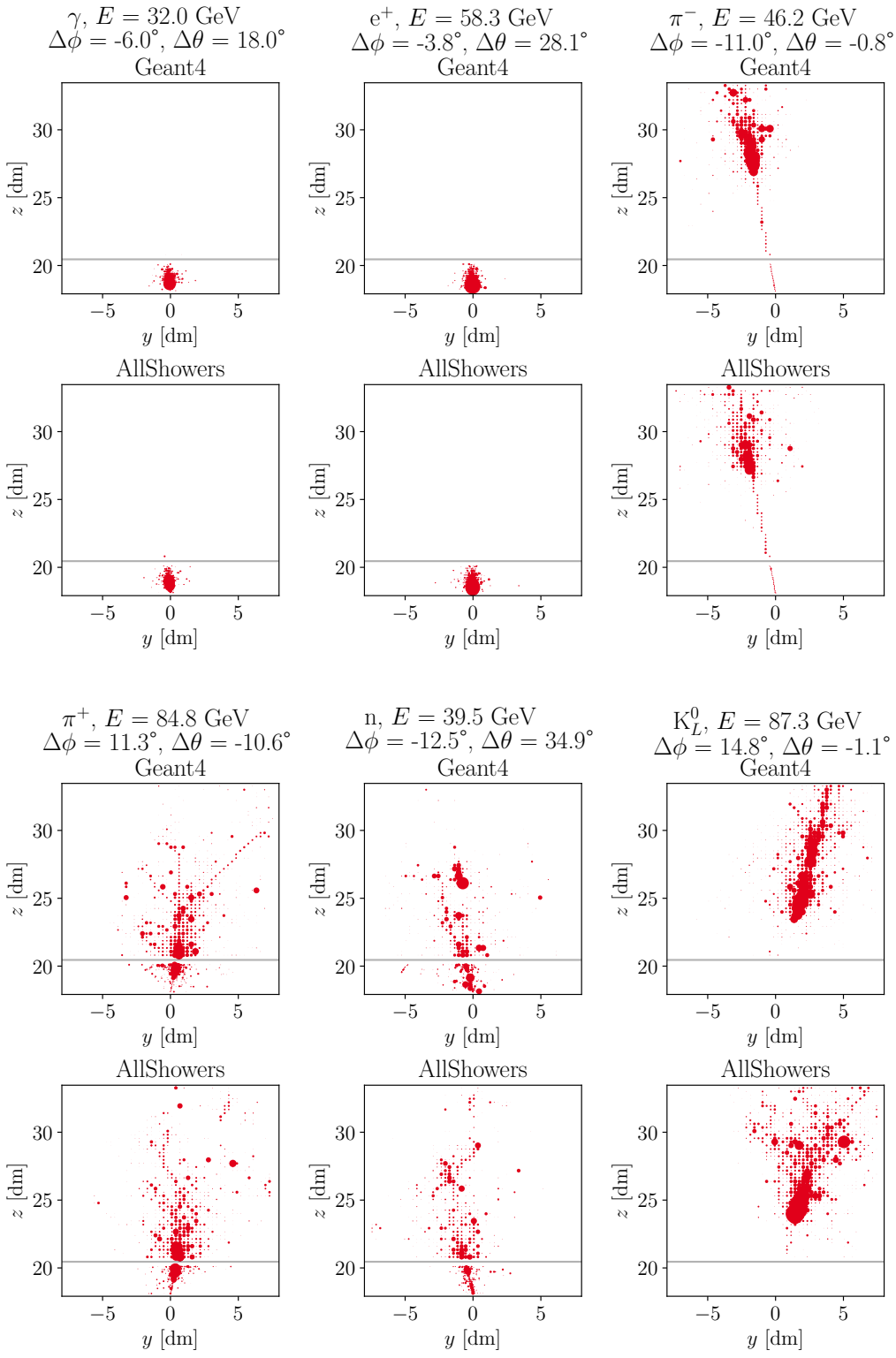

Figure 5: Comparison of individual showers simulated with Geant4 and with AllShowers for different incident particles, energies and angles. The point size indicates the energy of each hit. For these showers, the number of points per layer was taken from the Geant4 simulation rather than generated by the PointCountFM to allow for a more direct comparison of the spatial and energy distribution of hits.

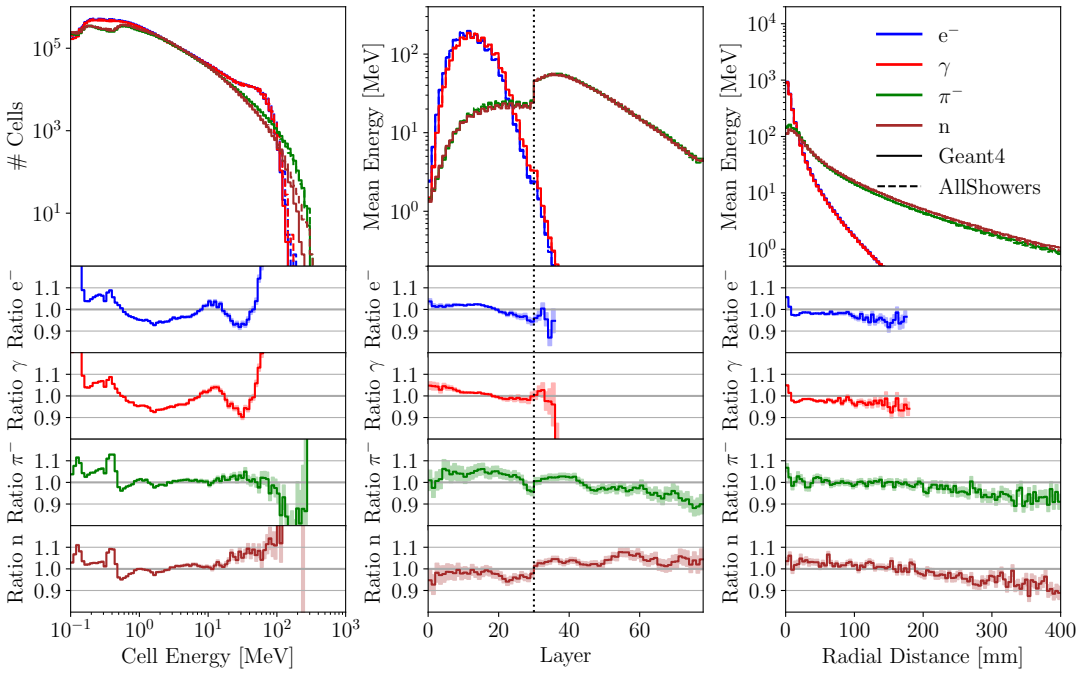

Figure 6: Histograms comparing Geant4 and AllShowers for different incident particle types and angles. The incident energy is fixed to 100 GeV. From left to right: cell energy spectrum, longitudinal energy distribution, and radial energy distribution. The solid lines represent Geant4 and the dashed lines AllShowers. The lower panels of each plot show the ratio of AllShowers over Geant4. For the cell energy spectrum, the Poisson error and for the longitudinal and radial energy distributions, the standard deviation of the mean is shown as error bars. Per particle type and generator, 10k showers were simulated/generated.

It is clear from these images that the embeddings (see section 3.3) used to encode the particle type are sufficient for AllShowers to produce appropriately tailored behavior in each shower. By eye, not only is each particle type distinct, but the features align well with those seen in the Geant4 simulation. The electromagnetic showers of the γ and e^+ each have the typical cloud-like distribution, well contained in the ECAL. The charged pions (π^+ and π^-) each have well defined MIP tracks in AllShowers, which correctly point to the start of the shower. For the π^-, this requires traversing right through the ECAL into the HCAL. Each pion shows a mild bend of the MIP track in opposite directions to account for the response to the magnetic field. Both pions then shower, with AllShowers displaying marginally fewer defined secondary tracks than Geant4, but still providing some, and displaying a very plausible shower pattern. The neutron (n) event produced by AllShowers also replicates the overall shower cone well, again perhaps showing fewer secondaries. Finally, as appropriate for a neutral particle, AllShowers does not generate a MIP track for the K_L^0 particle. The fetcher of neutral hadrons is is hidden for the neutron shower shown here since it starts showering immediately upon entering the calorimeter. The K_L^0 shower develops in AllShowers with good substructure, including visible internal secondary tracks, and a correct funnel shape. The shower start is marginally less aggressive in AllShowers than in Geant4, but it is a very plausible K_L^0 shower.

4.2 Distributions

While having visually credible individual showers is clearly an asset, almost all physics analysis happens on the ensemble-level. In figure 6, we present histograms comparing kinematic

behavior of Geant4 and AllShowers for selected particles; e^-, γ, π^+ and n. Other particles have similar accuracy, but are omitted for clarity in the plots.

In the top panels, both AllShowers (dashed line) and Geant4 (solid line) are shown for each of three quantities: cell energy spectrum, longitudinal energy distribution and radial energy distribution. For most values, the top plot shows no observable difference between Geant4 and AllShowers. In the lower plot, ratios of AllShowers to Geant4 are shown.

The most striking aspect of these plots is the clear dimorphism of electromagnetic showers (e^- and γ) and hadronic showers (π^+ and n). This bimodal behavior is well known, and that AllShowers accurately captures both variants demonstrates its flexibility.

In the cell energy spectrum on the left, AllShowers is within 10% of Geant4 for most of the range in all particle types. It makes a good replication of the MIP peak near 10^{-1} GeV, and does not significantly deviate until we reach the sparsely populated tails of the spectrum. The high energy tails tend to be somewhat overpopulated in AllShowers, the poor modeling is likely due to scarcity of this region in the training data.

In the longitudinal energy distribution in the centre, the same dimorphism between electromagnetic and hadronic showers is clear. AllShowers's behavior here is strongly influenced by the performance of the PointCountFM, and the agreement with Geant4 is within 10% for all but the extreme tails. This plot emphasizes the value of also modeling the HCAL for electromagnetic showers: γ showers in particular are not always well contained to the ECAL, and AllShowers manages to capture the tail that bleeds into the HCAL.

Finally, we see the dimorphism again in the radial distribution on the right. This radial distribution shows remarkably good agreement for the bulk of the shower. At the innermost core, some deviation is visible; but still within 10% of Geant4 for all particles. While there is more deviation in the tails, there are very few particles in these regions to work with, so it is expected that model performance may not be optimal here.

4.3 Comparison to CaloClouds3

For the case of photons only, we can compare the performance to the performance of the CaloClouds3 model [54]. CaloClouds3 is a fast generative diffusion model, specialized to only photon showers, trained on the ECAL only. As current generative models would not be applied in regions where different layer orientations meet, we also restrict the comparison data to photon showers, with $45° < \theta < 135°$ and $79° < \phi < 109°$. An energy range is chosen such that it sits comfortably inside both models training regions; 10 to 90 GeV.

In figure 7 the standard three kinematic profiles are shown for both fast models and Geant4. On the left, the cell energy spectrum for AllShowers is notably better aligned with Geant4 than CaloClouds3, in particular, AllShowers has a well formed replication of the MIP peak near 10^{-1} GeV. Neither model quite fits the high energy tail, but with very few data points, this is a challenging region to learn.

In the centre, CaloClouds3 and AllShowers perform equally well on the longitudinal energy distribution. CaloClouds3 is a little better at replicating the alternating layer pattern, but tends to overpopulate the start and end of the shower. On the right hand side of this plot, a grey band indicates the HCAL, for which only AllShowers has training data. This region is about as populated as the smallest bin in the ECAL, so its contribution is not negligible, and AllShowers's capacity to capture this information would be valuable in advanced reconstructions.

Finally, in the radial distribution on the right, AllShowers is significantly better than CaloClouds3. It maintains a flat ratio to Geant4 right out into a long distribution tail, and only marginally misrepresents the centre of the shower. CaloClouds3 is unable to keep a flat ratio, and deviates significantly from Geant4 towards the tail. While the deviation of CaloClouds3 in the tail here seems very large, it should be noted that the comparison of machine learning models ultimately will have to be done on physics observables, computed after a full event

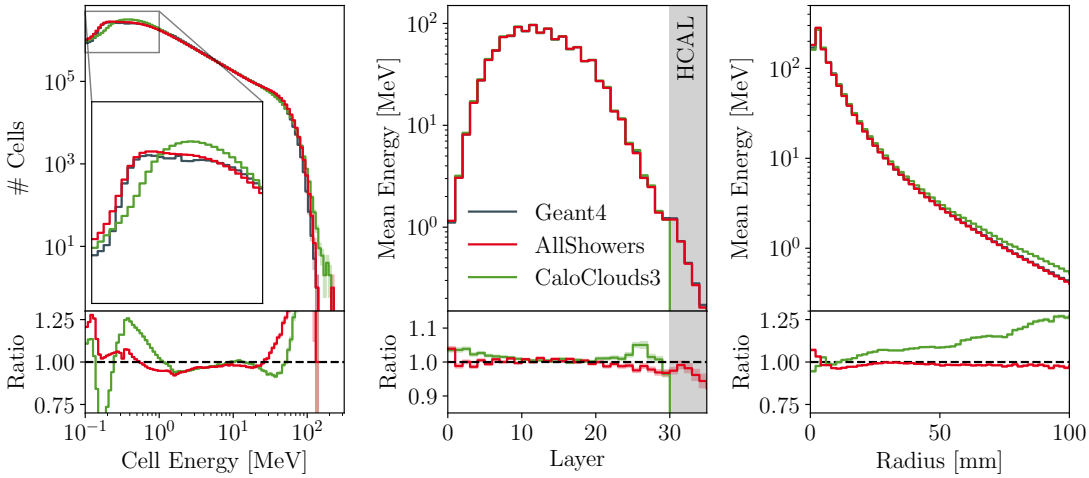

Figure 7: Comparison of AllShowers and CaloClouds3 on photon showers with incident energies uniformly distributed between 10 GeV and 90 GeV. Incident angles are distributed over the intersection of the respective training regions. From left to right: cell energy spectrum, longitudinal energy distribution and radial energy distribution. Per generator, 50k samples are used.

reconstruction has been applied, and that as shown in [61] the CaloClouds3 model performs reasonably well on π^0-reconstruction.

Conceptually, the key distinction between AllShowers and CaloClouds3 is that the diffusion model in CaloClouds3 generates points which are independent and identically distributed (iid). There are longitudinal correlations, imposed by the normalising flow component of CaloClouds3, but these are not known to the diffusion model, and they only describe the macro features, energy per layer and points per layer. This means that CaloClouds3 cannot capture point-to-point correlations. By contrast, AllShowers entertains correlations between points themselves. All these distributions demonstrate that even for a photon shower, the ability to capture subtle substructure can substantially improve the performance of the model.

The linearity of the reconstructed energy is always a key feature for a calorimeter, and must be well replicated in simulations. In figure 8 the linearity of photons as simulated by AllShowers and CaloClouds3 is plotted against a Geant4 reference. The simplified energy reconstruction is a linear sum of the energy deposits, with different scaling factors for sections of the calorimeter with different properties. Three scaling factors are chosen; one for the energy sum of the first 20 ECAL layers, then a second for energy sum of the last 10 ECAL layers, and finally a factor for the energy sum of the HCAL. All factors are chosen to minimize the mean squared error of the reconstructed Geant4 energies and then applied to both the Geant4 and the two ML model data.

In the reconstructed energy on the left AllShowers produces agreement with Geant4 on most points, however, some energies show significant deviations. AllShowers does not make energy predictions in PointCountFM, there is only a single energy correction factor applied, see section 3.7. This correction factor can raise or lower all points collectively, but cannot alter the relative height. By contrast, CaloClouds3 is performing very well across the whole range. Two elements contribute to this, the basic flat profile is achieved by the normalising flow in CaloClouds3, which predicts energy per layer for the model. Then in order to obtain the best mean value for all points, a single correction factor is applied, in the same way as for AllShowers.

For the energy resolution on the right, the range of reconstructed energies from AllShowers simulations is significantly too wide. This results in higher values (more variance) in the

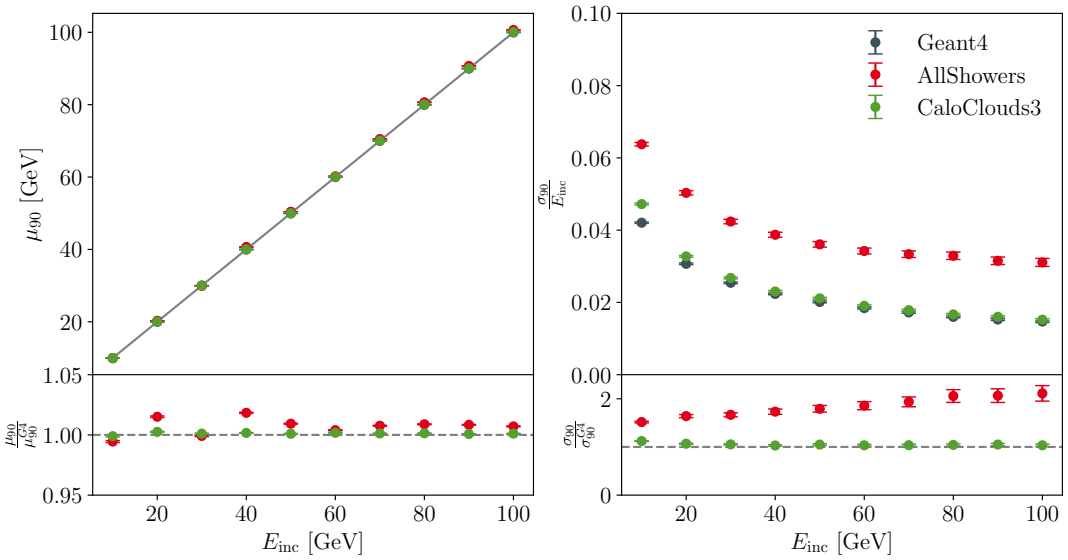

Figure 8: Linearity of rescaled energy sum for photon showers. Incident energies are chosen in steps of 10 GeV between 10 GeV and 100 GeV. Per energy and generator 10k samples are used. Incident angles are distributed over the intersection of the respective training regions.

resolution plot. By comparison, CaloClouds3 has a slight deviation in the lower energies, but is otherwise well matched to Geant4. The strong performance here is produced by the dedicated energy per layer predictions made by the normalising flow in CaloClouds3, which models the variations in energy accurately.

To improve the linearity and energy resolution of AllShowers it would be possible to add an energy per layer prediction to PointCountFM, in the same manner as is done in CaloClouds3. As AllShowers includes hadronic showers, it is desirable to retain correct energies for points in MIP tracks, and so a simple rescaling of the energy from PointCountFM would be detrimental. It is possible to conceive of various schemes that could rescale the energy per layer, while leaving the energy of MIPs intact, but we leave the exploration of these options to a future work.

4.4 Comparison to CaloHadronic

Another specialized model, which offers a comparison point for π^+ showers, is CaloHadronic [47]. CaloHadronic is trained only on π^+ that enter the calorimeter at a perpendicular angle, so both models will be asked for perpendicular incident angles. The energy range chosen is the full range that CaloHadronic was trained on: 10 to 90 GeV.

In figure 9 the standard three kinematic profiles are shown for both fast models and Geant4. On the left, in the cell energy spectrum, both fast models make a reasonably good approximation of the two MIP peaks (one in the ECAL and one in the HCAL). CaloHadronic significantly overestimates the high energy tail of the cell energy spectrum, while AllShowers manages to maintain a closer fit to Geant4 for significantly more of the distribution.

In the centre, the longitudinal energy distribution for AllShowers is notably better aligned with Geant4 than CaloHadronic. AllShowers can accurately capture the alternating layer pattern, and also shows better replication of the initial layers of the HCAL. Conditioning on the layer index and allowing to learn the layer properties in an embedding vector (see section 3.3) likely helps here. Overall, the longitudinal distribution created by AllShowers is remarkably well modelled.

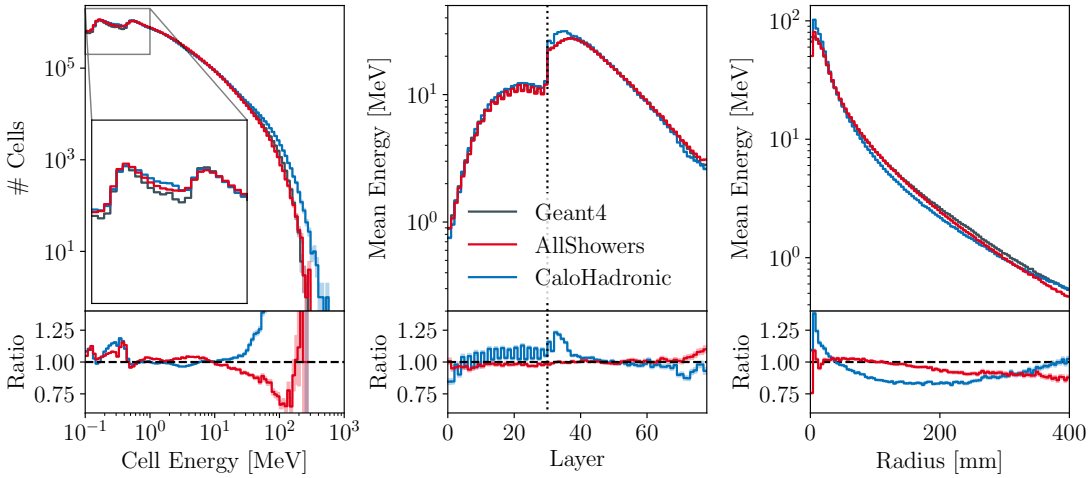

Figure 9: Comparison of AllShowers and CaloHadronic on π^+ showers with incident energies uniformly distributed between 10 GeV and 90 GeV. All pions enter the calorimeter perpendicularly. From left to right: cell energy spectrum, longitudinal energy distribution and radial energy distribution. Per generator, 50k samples are used.

Finally, on the right, we compare the radial distribution for AllShowers and CaloHadronic. In this distribution, CaloHadronic and AllShowers are more closely matched. CaloHadronic slightly overestimates the centre of the shower, and underestimates a significant portion of the bulk. AllShowers performs well in the centre, with only minor fluctuations in the first few bins, then tends to underestimate the tail.

Reconstructed energy of pions offers important insight into the relationship between energy deposits in the ECAL and HCAL for individual showers. In figure 10, we show the linearity and resolution of the reconstructed energy of π^+ showers generated by the two fast models and Geant4. The energy reconstruction is performed in the same way as for photons in section 4.3.

AllShowers offers a good reconstructed energy, marginally underestimating the energy of low energy π^+ showers, whereas CaloHadronic consistently overestimates the pion energy. Looking at the resolution of the reconstructed energies, neither model is performing well. In both cases, the distribution of the reconstructed energies is too wide at all incident energy points. AllShowers's performance is better, being within 50% of Geant4 for all incident energies, but both models leave a lot to be desired in this metric.

Overall, AllShowers clearly provides better kinematic descriptions of π^+ showers, with both the performance of the CNF-transformer in the radial direction and the combination of the PointCountFM and the CNF-transformer in the longitudinal direction demonstrating unprecedented accuracy on π^+ showers.

4.5 Timing

When comparing the timing of AllShowers to other models, we provide both the time for the execution of all 32 function evaluations used in the current version of the model, and a speculative time needed for a model with only 1 function evaluation. A reasonable future investigation for AllShowers would be to distill the model. A distilled model could require as little as a single function evaluation to attain similar performance, but at this point we have yet to achieve this optimization. So the timings for 32 function evaluations correspond to the current performance, and the timings for 1 function evaluation are speculative, but provide a good estimate of what timing performance might be attained by the next likely optimization.

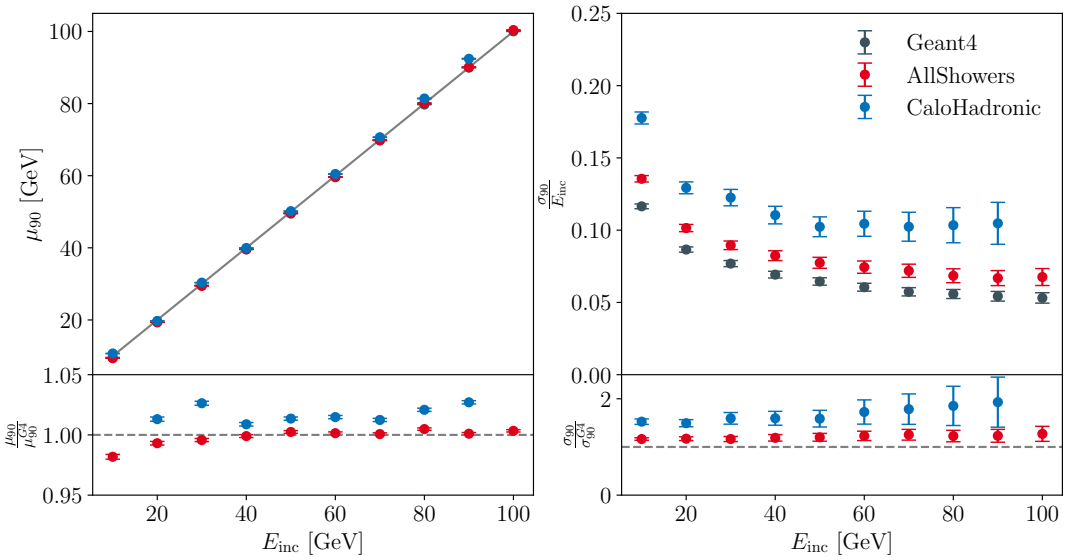

Figure 10: Linearity of rescaled energy sum for π^+ showers. Incident energies are chosen in steps of 10 GeV between 10 GeV and 100 GeV. Per energy and generator 10k samples are used. All pions enter the calorimeter perpendicularly.

To get the timing for 1 function evaluation, we generate showers with a single euler step.

In table 1, we compare the timing of AllShowers to CaloClouds3 and Geant4 on photon showers. Timings are measured for photons at 9 fixed incident energies; from 10 GeV to 90 GeV in steps of 10 GeV. The photons are all fired perpendicular to the calorimeter, and 100 batches are simulated per incident energy. Time of the first two batches is discarded, as the warm-up step may have longer, and more erratic timing dependant on memory allocation and "just in time" compilation. Times are only for the model's point generation itself, they do not include any overhead for moving data to or from the GPU or projecting hits into the detector geometry. This slightly complicates the comparison to Geant4, which by design places hits in sensitive cells only, and inherently incurs the overhead of the full detector geometry. To allow for a fair comparison, we force our models to use a single computational thread on CPU, as Geant4 does not support parallelism within a single event simulation.

All CPU timings are performed on an single core of an AMD EPYC 7402 processor with 512GB RAM. All GPU timings are performed on NVIDIA's A100.

CaloClouds3 is a fully distilled model, so 1 function evaluation is all that is ever used. It has also been aggressively optimized for the specific case of photons, including leveraging photon specific behaviors, such as the lack of significant substructure in the showers. With an iid assumption on the points, larger batch sizes become particularly efficient. AllShowers's current format prohibits specialized treatment of photons, and being the first generation of this model design, it has not undergone such significant optimization as CaloClouds3, so it is expected that AllShowers cannot compete in inference time with CaloClouds3. Indeed, CaloClouds3 is at least two orders of magnitude faster on CPU. On the GPU the difference is less dramatic, but overall it is seen that CaloClouds3 will remain significantly faster until AllShowers is distilled or otherwise optimized.

In table 2, a similar timing comparison is shown for pion showers. This time CaloHadronic is used as a comparison point, and for CaloHadronic, the NFE is also a tunable parameter. In the table, timings are shown for both a hypothetical distilled version with one function evaluation and for the number of function evaluations used in the current versions of the models. As with the photon timings, π^+ showers are all fired perpendicular to the calorimeter,

Hardware	Model	NFE	Batch Size	Time / Sample [s]	Speed-factor
CPU	Geant4	-	1	2.88	1.0x
	CaloClouds3	1	1	0.014	194.3x
			16	0.0041	654.5x
	AllShowers	1	1	0.17	16.7x
			16	0.16	17.6x
		32	1	5.0	0.6x
			16	5.1	0.6x
GPU	CaloClouds3	1	1	0.014	208.3x
			16	0.00088	3256. x
	AllShowers	1	1	0.014	209.3x
			16	0.0010	2806. x
		32	1	0.045	64.3x
			16	0.0050	581.6x

Table 1: Timing comparison between Geant4, CaloClouds3, and AllShowers on photon showers.

and 9 fixed energies are simulated between 10 and 100 GeV. Here the comparison is closer, as both models are compelled to deal with substructure in hadronic showers.

As CaloHadronic was a pilot model, designed to demonstrate the potential to combine ECAL and HCAL simulation, its code was never restructured to allow compilation. Thus, if CaloHadronic were timed including preprocessing of input data and postprocessing of generated outputs, it would be unrealistically slow. Instead, only the evaluation of the PyTorch model itself was timed, as this would dominate the timing in a more realistic deployment. Due to the omission of all other elements from the timing, the times for CaloHadronic can be regarded as mildly optimistic. Despite this, AllShowers comes out as faster than CaloHadronic, both at a single function evaluation, and with the NFE that is customary for the model. This shows all round more efficient use of resources, including good GPU performance.

5 Conclusion

We have presented AllShowers, a novel generative model for high-granularity calorimeter shower simulation. AllShowers is the a unified generative model capable of generating multiple particle types, encompassing both electromagnetic and hadronic showers, within a single architecture. This can help reduce the memory footprint, a significant bottleneck in large-scale Monte Carlo production, by allowing loading a single model for all particle types. Moreover, the model is conditioned on incident angle and energy, enabling broad applicability, and it can simultaneously simulate energy depositions across both ECAL and HCAL, thereby enabling end-to-end calorimeter response generation.

AllShowers shows strong agreement with Geant4 across a range of individual-shower features, including aspects of the fine spatial structure accessible with highly granular calorimeters, as well as for ensemble-level distributions spanning multiple particle species. In comparisons at the shower level, its performance is competitive with specialized baselines — CaloClouds3 for photons and CaloHadronic for pions — often yielding closer agreement on

Hardware	Model	NFE	Batch Size	Time / Sample [s]	Speed-factor
CPU	Geant4	-	1	2.09	1.0x
	CaloHadronic	1	1	0.59	3.5x
			16	0.73	2.8x
		59	1	34.8	< 0.1x
			16	43.3	< 0.1x
	AllShowers	1	1	0.12	16.7x
			16	0.12	18.0x
		32	1	3.5	0.6x
			16	3.6	0.6x
GPU	CaloHadronic	1	1	0.0086	243.0x
			16	0.0033	633.3x
		59	1	0.40	5.3x
			16	0.15	13.7x
	AllShowers	1	1	0.013	157.3x
			16	0.0010	1990.5x
		32	1	0.044	47.7x
			16	0.0047	447.7x

Table 2: Timing comparison between Geant4, CaloHadronic, and AllShowers on pion showers. Geant4 and CaloHadronic times are taken form [47].

several observables. For photons, the absence of an iid assumption leads to slower generation than CaloClouds3, while potentially capturing additional correlations in the shower development. However, a definitive assessment of the trade-off between computational performance and physics fidelity for these models ultimately requires evaluating realistic physics observables after full detector reconstruction. For pions, AllShowers achieves comparable or improved agreement relative to CaloHadronic, while also providing faster sampling.

Looking forward, we aim to improve the energy resolution of AllShowers either by generating layer-wise energy deposits similar to CaloClouds3 or by applying a postprocessing step. Additionally, we plan to distill the model to reduce the number of function evaluations (NFE) required at sampling time and to extend AllShowers to additional detector geometries, further broadening its applicability to high-energy physics simulation workflows.

Acknowledgements

We would like to thank Martina Mozzanica for providing shower data generated with CaloHadronic for comparison. We thank Anatolii Korol for helping us understand ddsim and the ILC Soft framework, for giving insight into how the training data for CaloHadronic was simulated, and for providing the regularized readout geometry created for earlier datasets. Furthermore, we would like to thank Joschka Birk for fruitful discussions, especially for suggesting that we explore the ranger and lookahead optimizers. We thank Thomas Madlener for providing valuable feedback on the manuscript.

Funding information This research was supported in part by the Maxwell computational resources operated at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany. This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No 101004761. We acknowledge support by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy – EXC 2121 Quantum Universe – 390833306 and via the KISS consortium (05D23GU4, 13D22CH5) funded by the German Federal Ministry of Research, Technology and Space (BMFTR) in the ErUM-Data action plan.

A Code and Data Availability

The code written for this work is available in the following Git repositories:

CNF-transformer: https://github.com/FLC-QU-hep/AllShowers

PointCountFM: https://github.com/FLC-QU-hep/PointCountFM

Ranger-Light optimizer: https://github.com/FLC-QU-hep/ranger-lite

Collection of shower IO utilities: https://github.com/FLC-QU-hep/ShowerData

The training datasets simulated for this work are available at:

AllShowers Dataset: https://doi.org/10.5281/zenodo.18020348

B Number of Trainable Parameters

Model	Layer-level model	Point-level model	Total
AllShowers	351,822	263,491	615,313
CaloClouds3	6,026,520	69,640	6,096,160
CaloHadronic	349,905	1,784,724	2,134,629

Table 3: Number of trainable parameters for AllShowers, CaloClouds3, and CaloHadronic.

In table 3, we compare the number of trainable parameters for AllShowers, CaloClouds3, and CaloHadronic. Shown are the number of parameters in the layer-level model (PointCountFM for AllShowers and CaloHadronic, and the normalizing flow for CaloClouds3), the point-level model (CNF-transformer for AllShowers, and the diffusion models for CaloClouds3 and CaloHadronic), and the total number of parameters. It is evident that AllShowers has a significantly smaller total number of parameters compared to both CaloClouds3 and CaloHadronic.

C Hyper-Parameters

All hyper-parameters used to train PointCountFM can be found in table 4 and those used to train the CNF-transformer in table 5.

Type	Parameter	Value
Data Preprocessing	incident particle type	one-hot encoding
	incident energy	Standard Scaling of E_{inc}
	incident angle	unit sphere representation
	point per layer	Standard Scaling of $\log(0.5 + N_i)$
Model	hidden layers	5
	hidden dims	128, 256, 512, 256, 128
	activation	ReLU
Training	optimizer	Adam
	learning rate scheduler	OneCycle
	maximum learning rate	10^{-3}
	batch size	1024
	epochs	1000
Sampling	ODE solver	Heun
	NFE	100

Table 4: Hyper-parameters used for the PointCountFM model.

Type	Parameter	Value
Data Preprocessing	point x, y	Standard Scaling
	point energy	Standard Scaling of $\log(E)$
	incident energy	Standard Scaling of $\log(E_{\text{inc}})$
	flow time	faure embedding with 3 frequencies
	incident angle	unit sphere representation
	OT mapping	layer-and-shower-wise
Model	embedding dim	64
	transformer encoder blocks	4
	attention heads	4
	feedforward dim	256
	attention masking	custom calorimeter-layer-based
Training	optimizer	Ranger (Lookahead + RAdam)
	learning rate scheduler	cosine annealing
	initial learning rate	10^{-3}
	weight decay	10^{-2}
	gradient clipping	0.2
	batch size	256
	epochs	200
Sampling	ODE solver	midpoint
	NFE	32

Table 5: Hyper-parameters used for the CNF-transformer model.

References

[1] J. Albrecht *et al.*, *A Roadmap for HEP Software and Computing R&D for the 2020s*, Comput. Softw. Big Sci. **3**(1), 7 (2019), doi:10.1007/s41781-018-0018-8, https://arxiv.org/abs/1712.06982.

[2] A. Boehnlein, C. Biscarat, A. Bressan, D. Britton, R. Bolton, F. Gaede, C. Grandi, F. Hernandez, T. Kuhr, G. Merino, F. Simon and G. Watts, *HL-LHC Software and Computing Review Panel, 2nd Report*, Tech. Rep. CERN-LHCC-2022-007, LHCC-G-183, CERN, Geneva (2022), https://cds.cern.ch/record/2803119.

[3] 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**(4), 042003 (2018), doi:10.1103/PhysRevLett.120.042003, https://arxiv.org/abs/1705.02355.

[4] 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**(1), 014021 (2018), doi:10.1103/PhysRevD.97.014021, https://arxiv.org/abs/1712.10321.

[5] L. de Oliveira, M. Paganini and B. Nachman, *Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters*, J. Phys. Conf. Ser. **1085**(4), 042017 (2018), doi:10.1088/1742-6596/1085/4/042017, https://arxiv.org/abs/1711.08813.

[6] 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**(1), 4 (2018), doi:10.1007/s41781-018-0008-x, https://arxiv.org/abs/1802.03325.

[7] M. Erdmann, J. Glombitza and T. Quast, *Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network*, Comput. Softw. Big Sci. **3**(1), 4 (2019), doi:10.1007/s41781-018-0019-7, https://arxiv.org/abs/1807.01954.

[8] P. Musella and F. Pandolfi, *Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks*, Comput. Softw. Big Sci. **2**(1), 8 (2018), doi:10.1007/s41781-018-0015-y, https://arxiv.org/abs/1805.00850.

[9] D. Belayneh *et al.*, *Calorimetry with deep learning: particle simulation and reconstruction for collider physics*, Eur. Phys. J. C **80**(7), 688 (2020), doi:10.1140/epjc/s10052-020-8251-9, https://arxiv.org/abs/1912.06794.

[10] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn, *GANplifying event samples*, SciPost Phys. **10**(6), 139 (2021), doi:10.21468/SciPostPhys.10.6.139, https://arxiv.org/abs/2008.06545.

[11] ATLAS collaboration, *Fast simulation of the ATLAS calorimeter system with Generative Adversarial Networks*, Tech. Rep. ATL-SOFT-PUB-2020-006, CERN, Geneva (2020), https://cds.cern.ch/record/2746032.

[12] A. Ghosh, *Deep generative models for fast shower simulation in ATLAS*, J. Phys. Conf. Ser. **1525**(1), 012077 (2020), doi:10.1088/1742-6596/1525/1/012077.

[13] G. Aad *et al.*, *AtlFast3: The Next Generation of Fast Simulation in ATLAS*, Comput. Softw. Big Sci. **6**(1), 7 (2022), doi:10.1007/s41781-021-00079-7, https://arxiv.org/abs/2109.02551.

[14] G. Aad *et al.*, *Deep Generative Models for Fast Photon Shower Simulation in ATLAS*, Comput. Softw. Big Sci. **8**(1), 7 (2024), doi:10.1007/s41781-023-00106-9, https://arxiv.org/abs/2210.06204.

[15] M. Faucci Giannelli and R. Zhang, *CaloShowerGAN, a generative adversarial network model for fast calorimeter shower simulation*, Eur. Phys. J. Plus **139**(7), 597 (2024), doi:10.1140/epjp/s13360-024-05397-4, https://arxiv.org/abs/2309.06515.

[16] E. Simsek, B. Isildak, A. Dogru, R. Aydogan, A. B. Bayrak and S. Ertekin, *CALPAGAN: Calorimetry for Particles Using Generative Adversarial Networks*, PTEP **2024**(8), 083C01 (2024), doi:10.1093/ptep/ptae106, https://arxiv.org/abs/2401.02248.

[17] 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* (2022), https://arxiv.org/abs/2211.15380.

[18] S. Hoque, H. Jia, A. Abhishek, M. Fadaie, J. Q. Toledo-Marín, T. Vale, R. G. Melko, M. Swiatlowski and W. T. Fedorko, *CaloQVAE: Simulating high-energy particle-calorimeter interactions using hybrid quantum-classical generative models*, Eur. Phys. J. C **84**(12), 1244 (2024), doi:10.1140/epjc/s10052-024-13576-x, https://arxiv.org/abs/2312.03179.

[19] 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* (2024), https://arxiv.org/abs/2405.06605.

[20] C. Krause and D. Shih, *Fast and accurate simulations of calorimeter showers with normalizing flows*, Phys. Rev. D **107**(11), 113003 (2023), doi:10.1103/PhysRevD.107.113003, https://arxiv.org/abs/2106.05285.

[21] C. Krause and D. Shih, *Accelerating accurate simulations of calorimeter showers with normalizing flows and probability density distillation*, Phys. Rev. D **107**(11), 113004 (2023), doi:10.1103/PhysRevD.107.113004, https://arxiv.org/abs/2110.11377.

[22] S. Schnake, D. Krücker and K. Borras, *Generating calorimeter showers as point clouds*, In *Machine Learning and the Physical Sciences, Workshop at the 36th conference on Neural Information Processing Systems (NeurIPS)* (2022), https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_77.pdf.

[23] C. Krause, I. Pang and D. Shih, *CaloFlow for CaloChallenge dataset 1*, SciPost Phys. **16**(5), 126 (2024), doi:10.21468/SciPostPhys.16.5.126, https://arxiv.org/abs/2210.14245.

[24] A. Xu, S. Han, X. Ju and H. Wang, *Generative machine learning for detector response modeling with a conditional normalizing flow*, JINST **19**(02), P02003 (2024), doi:10.1088/1748-0221/19/02/P02003, https://arxiv.org/abs/2303.10148.

[25] M. R. Buckley, C. Krause, I. Pang and D. Shih, *Inductive simulation of calorimeter showers with normalizing flows*, Phys. Rev. D **109**(3), 033006 (2024), doi:10.1103/PhysRevD.109.033006, https://arxiv.org/abs/2305.11934.

[26] I. Pang, D. Shih and J. A. Raine, *Calorimeter shower superresolution*, Phys. Rev. D **109**(9), 092009 (2024), doi:10.1103/PhysRevD.109.092009, https://arxiv.org/abs/2308.11700.

[27] F. Ernst, L. Favaro, C. Krause, T. Plehn and D. Shih, *Normalizing Flows for High-Dimensional Detector Simulations*, SciPost Phys. **18**, 081 (2025), doi:10.21468/SciPostPhys.18.3.081, https://arxiv.org/abs/2312.09290.

[28] S. Schnake, D. Krücker and K. Borras, *CaloPointFlow II Generating Calorimeter Showers as Point Clouds* (2024), https://arxiv.org/abs/2403.15782.

[29] H. Du, C. Krause, V. Mikuni, B. Nachman, I. Pang and D. Shih, *Unifying simulation and inference with normalizing flows*, Phys. Rev. D **111**(7), 076004 (2025), doi:10.1103/PhysRevD.111.076004, https://arxiv.org/abs/2404.18992.

[30] E. Majerz, W. Dzwinel and J. Kitowski, *Inverse Autoregressive Flows for Zero Degree Calorimeter fast simulation*, In *39th Annual Conference on Neural Information Processing Systems: Includes Machine Learning and the Physical Sciences (ML4PS)* (2025), https://arxiv.org/abs/2512.20346.

[31] J. Birk, F. Gaede, A. Hallin, G. Kasieczka, M. Mozzanica and H. Rose, *OmniJet-α_C: learning point cloud calorimeter simulations using generative transformers*, JINST **20**(07), P07007 (2025), doi:10.1088/1748-0221/20/07/P07007, https://arxiv.org/abs/2501.05534.

[32] V. Mikuni and B. Nachman, *Score-based generative models for calorimeter shower simulation*, Phys. Rev. D **106**(9), 092009 (2022), doi:10.1103/PhysRevD.106.092009, https://arxiv.org/abs/2206.11898.

[33] F. T. Acosta, V. Mikuni, B. Nachman, M. Arratia, B. Karki, R. Milton, P. Karande and A. Angerami, *Comparison of point cloud and image-based models for calorimeter fast simulation*, JINST **19**(05), P05003 (2024), doi:10.1088/1748-0221/19/05/P05003, https://arxiv.org/abs/2307.04780.

[34] V. Mikuni and B. Nachman, *CaloScore v2: single-shot calorimeter shower simulation with diffusion models*, JINST **19**(02), P02001 (2024), doi:10.1088/1748-0221/19/02/P02001, https://arxiv.org/abs/2308.03847.

[35] O. Amram and K. Pedro, *Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation*, Phys. Rev. D **108**(7), 072014 (2023), doi:10.1103/PhysRevD.108.072014, https://arxiv.org/abs/2308.03876.

[36] 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*, SciPost Phys. **18**(6), 195 (2025), doi:10.21468/SciPostPhys.18.6.195, https://arxiv.org/abs/2401.13162.

[37] D. Kobylianskii, N. Soybelman, E. Dreyer and E. Gross, *Graph-based diffusion model for fast shower generation in calorimeters with irregular geometry*, Phys. Rev. D **110**(7), 072003 (2024), doi:10.1103/PhysRevD.110.072003, https://arxiv.org/abs/2402.11575.

[38] C. Jiang, S. Qian and H. Qu, *BUFF: Boosted Decision Tree based Ultra-Fast Flow matching* (2024), https://arxiv.org/abs/2404.18219.

[39] L. Favaro, A. Ore, S. P. Schweitzer and T. Plehn, *CaloDREAM – Detector Response Emulation via Attentive flow Matching*, SciPost Phys. **18**, 088 (2025), doi:10.21468/SciPostPhys.18.3.088, https://arxiv.org/abs/2405.09629.

[40] J. Brehmer, V. Bresó, P. de Haan, T. Plehn, H. Qu, J. Spinner and J. Thaler, *A Lorentz-equivariant transformer for all of the LHC*, SciPost Phys. **19**(4), 108 (2025), doi:10.21468/SciPostPhys.19.4.108, https://arxiv.org/abs/2411.00446.

[41] 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**(1), 4916 (2024), doi:10.1038/s41467-024-49104-4, [Erratum: Nature Commun. 115, 5825 (2024)], https://arxiv.org/abs/2303.08046.

[42] F. Carminati, A. Gheata, G. Khattak, P. Mendez Lorenzo, S. Sharan and S. Vallecorsa, *Three dimensional Generative Adversarial Networks for fast simulation*, J. Phys. Conf. Ser. **1085**(3), 032016 (2018), doi:10.1088/1742-6596/1085/3/032016.

[43] 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**(3), 035044 (2023), doi:10.1088/2632-2153/acefa9, https://arxiv.org/abs/2303.18150.

[44] 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**(1), 13 (2021), doi:10.1007/s41781-021-00056-0, https://arxiv.org/abs/2005.05334.

[45] 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**, 03003 (2021), doi:10.1051/epjconf/202125103003, https://arxiv.org/abs/2102.12491.

[46] 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**(2), 025014 (2022), doi:10.1088/2632-2153/ac7848, https://arxiv.org/abs/2112.09709.

[47] T. Buss, F. Gaede, G. Kasieczka, A. Korol, K. Krüger, P. McKeown and M. Mozzanica, *CaloHadronic: a diffusion model for the generation of hadronic showers* (2025), https://arxiv.org/abs/2506.21720.

[48] S. Bieringer, A. Butter, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, B. Nachman, T. Plehn and M. Trabs, *Calomplification — the power of generative calorimeter models*, JINST **17**(09), P09028 (2022), doi:10.1088/1748-0221/17/09/P09028, https://arxiv.org/abs/2202.07352.

[49] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh and D. Shih, *L2LFlows: generating high-fidelity 3D calorimeter images*, JINST **18**(10), P10017 (2023), doi:10.1088/1748-0221/18/10/P10017, https://arxiv.org/abs/2302.11594.

[50] T. Buss, F. Gaede, G. Kasieczka, C. Krause and D. Shih, *Convolutional L2LFlows: generating accurate showers in highly granular calorimeters using convolutional normalizing flows*, JINST **19**(09), P09003 (2024), doi:10.1088/1748-0221/19/09/P09003, https://arxiv.org/abs/2405.20407.

[51] 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**(11), P11025 (2023), doi:10.1088/1748-0221/18/11/P11025, https://arxiv.org/abs/2305.04847.

[52] 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**(04), P04020 (2024), doi:10.1088/1748-0221/19/04/P04020, https://arxiv.org/abs/2309.05704.

[53] P. Raikwar, A. Zaborowska, P. McKeown, R. Cardoso, M. Piorczynski and K. Yeo, *A Generalisable Generative Model for Multi-Detector Calorimeter Simulation* (2025), https://arxiv.org/abs/2509.07700.

[54] T. Buss, H. Day-Hall, F. Gaede, G. Kasieczka, K. Krüger, A. Korol, T. Madlener, P. McKeown, M. Mozzanica and L. Valente, *CaloClouds3: Ultra-Fast Geometry-Independent Highly-Granular Calorimeter Simulation* (2025), https://arxiv.org/abs/2511.01460.

[55] F. Gaede, G. Kasieczka and L. Valente, *Cross-Geometry Transfer Learning in Fast Electromagnetic Shower Simulation* (2025), https://arxiv.org/abs/2512.00187.

[56] L. Favaro, A. Giammanco and C. Krause, *A universal vision transformer for fast calorimeter simulations* (2026), https://arxiv.org/abs/2601.05289.

[57] B. Hashemi and C. Krause, *Deep generative models for detector signature simulation: A taxonomic review*, Rev. Phys. **12**, 100092 (2024), doi:10.1016/j.revip.2024.100092, https://arxiv.org/abs/2312.09597.

[58] O. Amram *et al.*, *CaloChallenge 2022: a community challenge for fast calorimeter simulation*, Rept. Prog. Phys. **88**(11), 116201 (2025), doi:10.1088/1361-6633/ae1304, https://arxiv.org/abs/2410.21611.

[59] H. Abramowicz *et al.*, *International Large Detector: Interim Design Report* (2020), https://arxiv.org/abs/2003.01116.

[60] J. Repond *et al.*, *Design and Electronics Commissioning of the Physics Prototype of a Si-W Electromagnetic Calorimeter for the International Linear Collider*, JINST **3**, P08001 (2008), doi:10.1088/1748-0221/3/08/P08001, https://arxiv.org/abs/0805.4833.

[61] T. Buss, H. Day-Hall, F. Gaede, G. Kasieczka, K. Krüger, A. Korol, T. Madlener and P. McKeown, *A First Full Physics Benchmark for Highly Granular Calorimeter Surrogates* (2025), https://arxiv.org/abs/2511.17293.

[62] The Geant4 Collaboration, *Geant4–a simulation toolkit*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506**(3), 250 (2003), doi:10.1016/S0168-9002(03)01368-8.

[63] M. Frank, F. Gaede, C. Grefe and P. Mato, *Dd4hep: A detector description toolkit for high energy physics experiments*, Journal of Physics: Conference Series **513**(2), 022010 (2014), doi:10.1088/1742-6596/513/2/022010.

[64] R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud, *Neural Ordinary Differential Equations* (2018), https://arxiv.org/abs/1806.07366.

[65] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel and M. Le, *Flow matching for generative modeling* (2023), https://arxiv.org/abs/2210.02747.

[66] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron and R. Ng, *Fourier features let networks learn high frequency functions in low dimensional domains* (2020), https://arxiv.org/abs/2006.10739.

[67] L. van der Maaten, G. Hinton and Y. Rachmad, *Viualizing data using t-sne*, Journal of Machine Learning Research **9**, 2579 (2008).

[68] J. Dong, B. Feng, D. Guessous, Y. Liang and H. He, *Flex attention: A programming model for generating optimized attention kernels* (2024), https://arxiv.org/abs/2412.05496.

[69] R. Child, S. Gray, A. Radford and I. Sutskever, *Generating long sequences with sparse transformers* (2019), https://arxiv.org/abs/1904.10509.

[70] I. Beltagy, M. E. Peters and A. Cohan, *Longformer: The long-document transformer* (2020), https://arxiv.org/abs/2004.05150.

[71] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang and A. Ahmed, *Big bird: Transformers for longer sequences* (2021), https://arxiv.org/abs/2007.14062.

[72] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention Is All You Need*, In *31st International Conference on Neural Information Processing Systems* (2017), https://arxiv.org/abs/1706.03762.

[73] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia *et al.*, *PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation*, In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, doi:10.1145/3620665.3640366 (2024).

[74] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf and Y. Bengio, *Improving and generalizing flow-based generative models with minibatch optimal transport* (2024), https://arxiv.org/abs/2302.00482.

[75] R. Flamary, C. Vincent-Cuaz, N. Courty, A. Gramfort, O. Kachaiev, H. Quang Tran, L. David, C. Bonet, N. Cassereau, T. Gnassounou, E. Tanguy, J. Delon *et al.*, *POT Python Optimal Transport*, https://github.com/PythonOT/POT.

[76] M. R. Zhang, J. Lucas, G. Hinton and J. Ba, *Lookahead optimizer: k steps forward, 1 step back* (2019), https://arxiv.org/abs/1907.08610.

[77] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization* (2017), https://arxiv.org/abs/1412.6980.

[78] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao and J. Han, *On the variance of the adaptive learning rate and beyond* (2021), https://arxiv.org/abs/1908.03265.

[79] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization* (2019), https://arxiv.org/abs/1711.05101.

[80] L. Wright, *Ranger - a synergistic optimizer.* (2019), https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer.