# SPADE: Split-and-Delay Embeddings for Autoregressive High-Granularity Calorimeter Simulation

Joschka Birk[1], Frank Gaede[2], Anna Hallin[1], Gregor Kasieczka[1], Martina Mozzanica[1], Henning Rose[1,*]

[1]Institute for Experimental Physics, Universität Hamburg, Luruper Chaussee 149, 22761 Hamburg, Germany
[2]Deutsches Elektronen-Synchrotron DESY, Notkestr. 85, 22607 Hamburg, Germany
*Author to whom any correspondence should be addressed.

**E-mail:** henning.rose@uni-hamburg.de

**Keywords:** Autoregressive Generation, Point Cloud Generation, Tokenization, Calorimeter Simulation, Detector Simulation

arXiv:2606.11304v1 [physics.ins-det] 9 Jun 2026

## Abstract

We introduce SPADE (SPlit And Delay Embeddings), an autoregressive transformer for sequences whose tokens carry multiple features. Rather than embedding these features jointly, SPADE embeds them independently. Delaying each feature stream relative to the previous one allows intra-token correlations to be learned by the standard self-attention mechanism. Applied to point-cloud calorimeter shower generation in the highly granular ILD detector, SPADE is competitive with the state of the art AllShowers model on photon showers, and substantially outperforms its VQ-VAE-based predecessor OmniJet-$\alpha_C$. The mechanism is applicable to any generative task with multi-feature tokens, enabling LLM-style pretraining workflows for higher-dimensional data.

## 1 Introduction

Current and future collider experiments in high-energy physics (HEP) require an increasing amount of Monte Carlo (MC) samples. Tools like GEANT4 [1] produce high-fidelity detector simulations, but require large amounts of compute resources, particularly for calorimeters. It is expected that the available resources will not be able to keep up with the demand [2, 3]. To avoid this becoming a bottleneck, there is a high interest in the development of generative machine learning (ML) methods that can supplement the MC samples. Existing methods include generative adversarial networks (GANs) [4–17], variational autoencoders (VAEs) [15, 18–20] and diffusion and flow models [21–41].

While most research into generative ML has focused on dedicated models, foundation models (FM) [42] with generative capabilities could provide another opportunity. Most foundation models built for HEP so far focus on jets and use some type of fill-in-the-blank pretraining target [43]. The model OmniJet-$\alpha$ [44], originally developed for jets, uses a next token prediction target similar to GPT [45]. This means that the pretraining itself results in a generative model. Since the input features are continuous, a VQ-VAE [46–48] converts them into discrete tokens to serve as targets for next token prediction. We have previously successfully adapted OmniJet-$\alpha$ to generate calorimeter showers, resulting in the model OmniJet-$\alpha_C$ [49]. Instead of the jet constituent features of OmniJet-$\alpha$, the input to this model consists of the spatial coordinates $(x, y, z)$ of the shower deposits, as well as the deposited energy $E$. Just as OmniJet-$\alpha$, OmniJet-$\alpha_C$ uses a VQ-VAE to tokenize the input features, replacing each $(x, y, z, E)$ input vector with a single integer (token). While this is a straightforward approach following the initial implementation of OmniJet-$\alpha$, it is not the most effective. In contrast to the fully continuous features of the jet constituents of the original model, which can't immediately be used as targets for next token prediction, three of the four calorimeter features are discrete. It should therefore in principle be possible to use these as-is. Furthermore, it is difficult to scale OmniJet-$\alpha_C$ to higher granularities: the higher the resolution needs to be, the more tokens the VQ-VAE needs. This affects not only the VQ-VAE itself, but also the main model, as the tokens need to be mapped to the embedding space and then back. Increasing the number of possible tokens in this mapping leads to a steep rise in the number of model parameters, resulting in increased training and inference times.

This work introduces SPADE: Split And Delay Embeddings for arbitrary detector granularities. SPADE *splits* the four hit features (three spatial coordinates and the deposited energy) into independent prediction streams, replacing the single VQ-VAE token of OmniJet-$\alpha_C$. This has two benefits: first, eliminating the VQ-VAE also eliminates the information loss inherent to VQ-VAE; second, using four separate prediction streams reduces the spatial vocabulary from cubic to linear in the per-axis grid resolution. Predicting the four features

independently would, however, lead to the correlations between them being lost. SPADE preserves these by staggering the streams along the sequence with progressively increasing *delays*. Instead of generating a hit all at once, the model builds each hit sequentially, allowing the autoregressive conditioning to maintain the crucial correlations between a hit's features.

Several recent approaches address related tokenization strategies. Ref. [50] also uses autoregressive token generation for calorimeter showers but keeps a 3D voxel index, scaling as $\mathcal{O}(N^3)$. Nested-GPT [41] factorizes per-token features but recovers their correlations through an inner recurrent decoder, whereas SPADE uses a delay along the main sequence. A similar delay mechanism has been applied earlier in audio generation to parallel residual-VQ codebooks [51], meaning that different representations of the same waveform functioned as different feature streams. However, our delayed features are not different representations of the same object, but distinct physical coordinates of a single hit.

The model is evaluated using two photon shower datasets: GettingHigh (irregular grid) and GettingSquare (regular grid, multiple granularities). In addition to a comparison with OmniJet-$\alpha_C$, we compare the performance of SPADE to a baseline model we refer to as *Combined*. This model removes the need for a VQ-VAE, but uses a single combined spatial vocabulary. We show that SPADE achieves superior performance to both OmniJet-$\alpha_C$ and the Combined baseline across photon showers. Furthermore, while both SPADE and the Combined baseline eliminate the lossy VQ-VAE of OmniJet-$\alpha_C$, SPADE uniquely avoids the Combined model's poor parameter scaling by reducing the parameter count by orders of magnitude. On the irregular GettingHigh dataset, SPADE is competitive with the state of the art AllShowers [52] reference model on most observables. On the regular GettingSquare dataset, the factorized SPADE embedding shows particularly clear advantages over Combined on observables that probe the spatial structure of the shower. Crucially, SPADE scales linearly rather than cubically with detector granularity, making it a practical architecture for the highly granular calorimeters of future collider experiments.

This paper is organized as follows. Section 2 describes the dataset used for training and evaluation. Section 3 introduces the three autoregressive architectures compared in this work: OmniJet-$\alpha_C$, Combined, and SPADE, as well as the flow matching AllShowers reference model. Section 4 presents the results: a four-way model comparison on the GettingHigh dataset with a realistic, irregular cell geometry (Section 4.1); and an evaluation of physics performance and parameter scaling on the regular GettingSquare dataset (Section 4.2). Additionally we show the superior training and representation efficiency in Section 5, and evaluate the inference time in Section 6. Section 7 summarizes the findings and discusses the broader applicability of the split-and-delay mechanism.

## 2 Dataset

All photon datasets used in this work derive from the same Geant4 [1] simulation campaign, originally produced for Ref. [53] using a detailed DD4hep [54] model of the Si-W electromagnetic calorimeter of the International Large Detector (ILD) [55, 56]. Single photons with energies uniformly distributed between 10 and 100 GeV enter the calorimeter perpendicularly at $z = 0$ and travel along the $z$-axis.

Geant4 records each individual energy deposition as a *step*; a *hit* is the sum of all steps that fall into a single voxel of a chosen detector grid. We use two datasets, GettingHigh and GettingSquare, that share the same underlying Geant4 steps but bin them onto different voxel grids.

GettingHigh **(irregular baseline).** The GettingHigh dataset maps each Geant4 step onto its physical sensor in a $30 \times 30 \times 30$ array; each non-empty sensor becomes one hit. Because the realistic ECal barrel prototype has slightly staggered layers, this one-to-one mapping inherits the irregular geometry of the real detector rather than imposing a regular grid, as illustrated in Fig. 1. The dataset contains 950 000 showers, split into 760 000 for training, 95 000 for validation and 95 000 for testing.

GettingSquare **(regular grid variants).** To study the impact of transverse granularity in a controlled setting, we first bin the Geant4 steps onto the finest grid, **x16** ($120 \times 120 \times 30$). The coarser variants are then obtained by reassigning each x16 hit to its enclosing voxel on a $60 \times 60 \times 30$ (**x4**) or $30 \times 30 \times 30$ (**x1**) grid, without merging, so a single coarse voxel can hold multiple hits inherited from x16. The longitudinal segmentation is fixed at 30 layers throughout, isolating transverse granularity as the single experimental variable. This construction is summarized in Fig. 2. Note that GettingSquare-x1 and GettingHigh share the same nominal voxel count but differ in geometry: GettingSquare-x1 imposes a regular grid, while GettingHigh preserves the irregular detector layout. We additionally use an **x1-merged** variant in which hits sharing a voxel are summed, mimicking a detector with coarser readout cells. Each GettingSquare dataset contains 750 000 showers, split 495 000 / 130 000 / 125 000 for training, validation and testing.

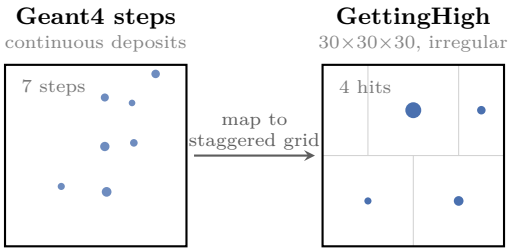



**Figure 1.** Mapping of continuous Geant4 shower deposits (left) to the GettingHigh representation (right). While the original Geant4 steps record exact spatial coordinates, GettingHigh bins these continuous deposits into a nominal $30 \times 30 \times 30$ geometry by mapping them onto physical, layer-staggered ILD sensors (visible as the offset between the upper and lower row dividers). Marker size in the GettingHigh panel scales with the per-sensor summed energy.

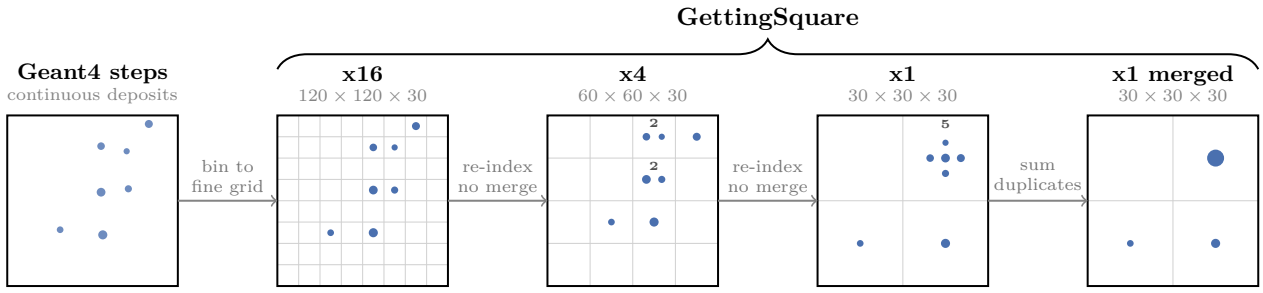



**Figure 2.** Construction of the GettingSquare datasets. The same Geant4 energy deposits (steps; leftmost panel) are binned onto a fine regular $120 \times 120 \times 30$ grid (GettingSquare-x16). The coarser GettingSquare-x4 and GettingSquare-x1 representations are obtained by reassigning each hit to its enclosing voxel at lower transverse resolution, without merging, so all three share an identical hit list, but several hits may occupy the same coarse voxel (numerical labels give the number of duplicate hits per shared cell). The GettingSquare-x1 merged-hit variant additionally sums the energies of duplicate hits into a single hit per cell (enlarged marker).

## 3 Methods

The autoregressive models considered in this work, OmniJet-$\alpha_C$, Combined and SPADE, are all decoder-only transformers that act on showers represented as sequences of hits. We define a shower as a collection of $n$ hits, $H = \{h_1, \ldots, h_n\}$, where each hit $h_i$ is characterized by its discrete spatial detector coordinates $(x_i, y_i, z_i)$ and its continuous deposited energy $E_i$. The sequence is zero-padded to a maximum length. After mapping the hits to a sequence of tokens, a causal mask is applied so that a token at position $i$ can only attend to tokens at positions $j \leq i$, allowing parallel training on next token prediction. At inference, showers are generated autoregressively, one hit at a time. The three models differ in (i) how the hits are ordered, (ii) how each hit is mapped to one or more tokens, and (iii) how the end of the sequence is determined, as detailed below. A schematic comparison of the three architectures is shown in Fig. 3.

### *3.1* OmniJet-$\alpha_C$

OmniJet-$\alpha_C$ uses a frozen VQ-VAE to map each four-dimensional hit $(x_i, y_i, z_i, E_i)$ to a single integer token drawn from a learned codebook of size $V = 65\,536$. The transformer is trained as a $V$-way classifier on next token prediction. After generation, the VQ-VAE decodes the tokens back to physical hit features.

The hits are ordered by deposited energy in descending order. Once tokenized, a special *start token* is added to the start of the sequence, and a *stop token* to the end. During generation, the model will sample autoregressively until it either generates the stop token or reaches the maximum sequence length.

Two limitations of this design motivate the present work. First, the embedding and unembedding layers contribute $2 \cdot V \cdot d$ parameters, where $d$ is the embedding dimension. With $V = 65{,}536$ and $d = 256$, these two layers alone account for 33.5M of the model's 36M parameters. This increase of the model size leads to slow training times, and scaling $V$ up further to accommodate finer detector granularity quickly becomes prohibitive. Second, the lossy compression of the VQ-VAE leads to spatial and energetic artifacts. Both Combined and SPADE address these issues by replacing the VQ-VAE with a direct embedding scheme, as outlined below.

### *3.2 Combined*

Since the spatial coordinates are already on a discrete grid, it should in principle be possible to use these coordinates directly without tokenization. The Combined model thus removes the VQ-VAE while still using a

single *combined* spatial vocabulary. The detector volume is treated as a regular grid of size $N_x \times N_y \times N_z$, and each hit position is mapped to a single scalar ID,

$$\mathrm{ID} = N_y N_z\, x + N_z\, y + z. \tag{1}$$

The vocabulary therefore scales multiplicatively with detector granularity, $V = N_x \cdot N_y \cdot N_z$. Unlike OmniJet-$\alpha_C$, energy is not encoded in the spatial token but treated as a separate continuous input. Instead of having a single token prediction head, Combined has one head for predicting the spatial ID, and a mixture-of-Gaussians head (Section 3.4) for predicting the continuous hit energy (see also Fig. 3). The continuous hit energy $E_i$ and the global incident energy $E_{\mathrm{inc}}$ are each passed through a linear projection to 16-dimensional vectors, whereas the dimensionality of the spatial ID embedding is 256. The three components are concatenated and projected to the 256-dimensional input expected by the transformer.

The spatial ID and the hit energy of the same hit are correlated. To preserve this correlation while still allowing the spatial and energy heads to predict their respective quantities independently, we delay the energy projection by one step relative to the spatial ID. The input at sequence position $i$ is therefore

$$T_i = \mathrm{Concat}(\mathrm{Emb}(\mathrm{ID}_i),\, \mathrm{Proj}_1(E_{i-1}),\, \mathrm{Proj}_2(E_{\mathrm{inc}}))\,, \tag{2}$$

so that by the time the model is about to predict $E_i$, it has already seen (and therefore becomes conditioned on) $\mathrm{ID}_i$. The empty slots created by the delay are padded as follows: missing spatial IDs are filled with a dedicated *missing ID* embedding, while missing energy projections are filled with zeros. These padding positions are masked out of the loss so that they do not contribute a training signal. The incident energy $E_{\mathrm{inc}}$ is concatenated to every token, as we found this to strengthen the conditioning and improve fidelity.

Hits are ordered first by longitudinal layer ($z$ ascending), and within each layer by deposited energy ($E$ descending). This layer-by-layer ordering provides a physically meaningful inductive bias that mirrors how showers actually develop. The layer-first ordering is used consistently across all Combined and SPADE runs.

### *3.3 SPADE*

SPADE replaces the single combined vocabulary with a *factorized* one. Rather than embedding a 3D voxel index, the three spatial coordinates $x$, $y$, and $z$ are embedded independently into 64-dimensional vectors, reducing the vocabulary scaling from multiplicative to additive: $V = N_x + N_y + N_z$. The single spatial ID prediction head from Combined now becomes three separate heads, one for each spatial coordinate, as illustrated in Fig. 3. We found that without normalization, the relative scale of the three spatial embeddings drifted during training and destabilized convergence. We therefore apply a LayerNorm [57] independently to each spatial embedding before they are combined. The continuous hit energy is predicted by the same mixture-of-Gaussians head as in Combined, described in Section 3.4.

SPADE generalizes the delay mechanism of Combined to the factorized setting. We stagger all four components ($x$, $y$, $z$, $E$) along the sequence with progressively increasing delays, so that the input at sequence position $i$ contains components belonging to four different hits:

$$T_i = \mathrm{Concat}(\mathrm{Emb}_{\mathrm{z}}(z_i),\, \mathrm{Emb}_{\mathrm{x}}(x_{i-1}),\, \mathrm{Emb}_{\mathrm{y}}(y_{i-2}),\, \mathrm{Proj}_1(E_{i-3}),\, \mathrm{Proj}_2(E_{\mathrm{inc}}))\,. \tag{3}$$

Each coordinate of a hit is now predicted at a different sequence position, so the heads condition on each other implicitly through the autoregressive structure. By the time $x_i$ is to be predicted, it has already seen (and therefore becomes conditioned on) $z_i$. In the same way, $y_i$ is conditioned on $z_i$ and $x_i$, and $E_i$ is conditioned on all spatial coordinates for position $i$.

The empty slots at the start and end of the sequence are padded and masked out of the loss exactly as in Combined. $E_{\mathrm{inc}}$ is appended to every token, and the concatenated input of Eq. (3) is linearly projected to the 256-dimensional transformer input.

### *3.4 Energy head*

Both Combined and SPADE predict the continuous hit energy $E_i$ through mixture-of-Gaussians (MoG) head [58] that parametrizes its conditional density as a mixture of $K$ Gaussians:

$$p(E_i) = \sum_{k=1}^{K} \pi_k\, \mathcal{N}(E_i \mid \mu_k, \sigma_k^2), \tag{4}$$

where the mixture weights $\pi_k$, means $\mu_k$ and variances $\sigma_k^2$ are produced at each sequence position by a small MLP sharing its inputs with the spatial heads. The spatial coordinates of the same hit are already encoded

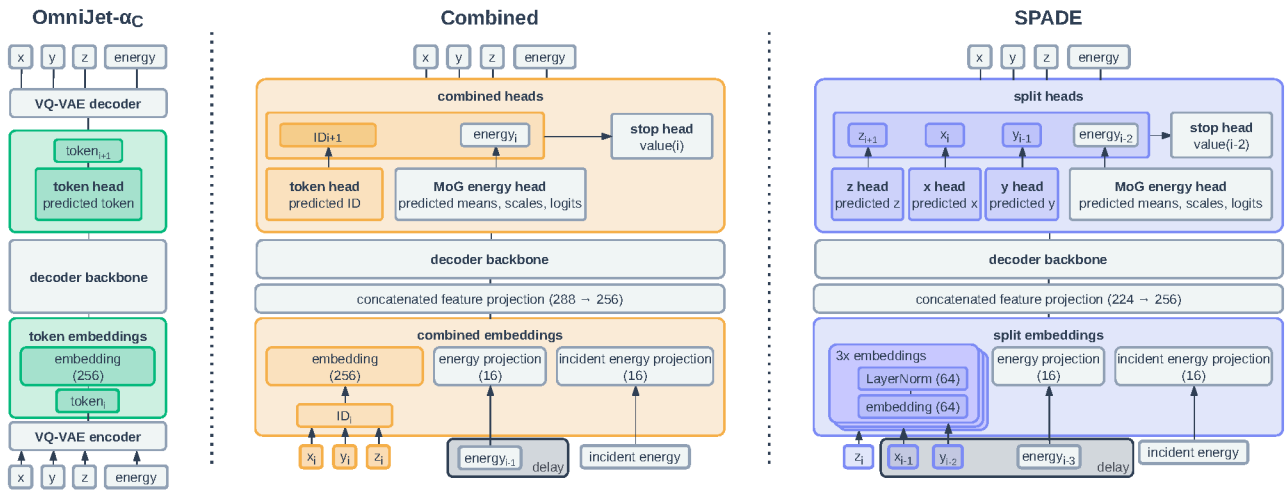



**Figure 3.** Schematic comparison of the model architectures. **Left:** OmniJet-$\alpha_C$ utilizes a VQ-VAE for continuous-to-discrete tokenization of spatial and energy features. **Middle:** The Combined baseline embeds joint 3D spatial coordinates into a single ID, delaying only the hit energy ($E_{i-1}$) to condition the next prediction. **Right:** SPADE replaces the joint vocabulary with factorized, independent spatial embeddings stabilized by LayerNorm. SPADE generalizes the delay mechanism by staggering all components ($z_i$, $x_{i-1}$, $y_{i-2}$, $E_{i-3}$) to enforce autoregressive conditioning among intra-hit components. Both Combined and SPADE utilize a Mixture of Gaussians (MoG) head for continuous energy prediction alongside a stop head for sequence termination.

in the transformer's hidden state at that position through the staggered input of Eqs. (2) and (3) and the causal self-attention, so the energy prediction is implicitly conditioned on $(x_i, y_i, z_i)$. We use $K = 4$ throughout. Because hits within a layer are ordered by descending energy, the energy $E_{i-1}$ is often numerically very close to the target $E_i$. To prevent the model from exploiting this ordering by trivially copying the previous input rather than learning the true distribution, we inject relative Gaussian noise into the continuous energy inputs during training. The head is trained by minimizing the negative log-likelihood of the predicted energy distributions, summed over non-padded positions. At inference, energies are sampled from the predicted mixture.

### *3.5 Stop head*

Combined and SPADE could in principle use an OmniJet-$\alpha_C$-like *stop token*, but the staggered input scheme makes this awkward to implement consistently. We instead introduce a dedicated stop head, which has the additional benefit of giving both models the same termination mechanism and enabling a clean head-to-head comparison. The stop head is a lightweight binary classifier with two output tokens, *continue* and *stop*, where the former allows shower token generation to proceed and the latter terminates it. Crucially, the stop head does not take the output of the backbone as input, but only the current shower state as predicted by the $x$, $y$, $z$, and $E$ heads. Incorporating the backbone output was found to significantly degrade results. The stop head is trained in a second stage with the backbone frozen, after which the final checkpoint is selected from the converged epochs by matching the generated $n_{\text{hits}}$ distribution to Geant4 on the validation set, with priority given to avoiding outlier showers of implausible length. Despite this tuning, we observe a specific failure mode during inference for SPADE where the model occasionally halt generation prematurely, leading to a severe under-prediction of the total deposited energy. This behavior affects around 0.35% of raw SPADE showers. To mitigate this, sampled showers are filtered through a short postprocessing step to reject these early-stopping failures. This mitigation strategy is described in detail in Appendix B, and all physics results in this paper are reported after this postprocessing is applied.

### *3.6 Decoder backbone*

Combined and SPADE utilize the same decoder architecture, featuring several upgrades over the OmniJet-$\alpha_C$ baseline. Relative sequence positions are encoded using Rotary Position Embedding (RoPE) [59]. To maximize training efficiency and generation throughput, the attention mechanism utilizes Multi-Query Attention [60] and hardware-optimized FlashAttention kernels [61, 62], alongside key-value (KV) caching during autoregressive inference.

### *3.7 Training*

Both Combined and SPADE are trained with the Ranger optimizer (Lookahead [63] + RAdam [64]) and constant learning-rate with linear warmup. The total loss combines the cross-entropy or negative log-likelihood loss from each prediction head. We observed that adjacent training checkpoints can differ noticeably in the quality of generated showers even when their validation losses are essentially identical; to suppress this checkpoint-to-checkpoint variability, sampling uses an exponential moving average (EMA) [65] of the training weights rather than the weights of any individual checkpoint. The full set of hyperparameters is listed in Appendix A.

### *3.8* AllShowers

One of the current state of the art approaches for generative modeling of calorimeter showers is the AllShowers framework [52]. This model provides a unified architecture designed to learn shower development across different particle types, incident angles, and incoming energies within a single conditional generative model. AllShowers consists of two models: the PointCountFM, which predicts the number of points per layer and directly determines the $z$ coordinate of each point, and a CNF-transformer, which predicts the remaining features ($x$, $y$ and energy $E$). The $z$ coordinate is deterministically assigned from the layer index, reducing the generative task of the CNF-transformer to three dimensions.

In this work, we train the AllShowers model on the photon datasets described in Section 2 (GettingHigh and GettingSquare). We remove the conditioning on particle type and incident angle, as both are redundant in this study, leaving the incident energy as the only conditioning feature. The input layer is resized accordingly; the rest of the architecture is unchanged.

## 4 Results

In the following we show the results grouped by the datasets the models were trained on. In Section 4.1 we compare OmniJet-$\alpha_C$, Combined, SPADE and AllShowers on the irregular GettingHigh dataset, whereas Section 4.2 shows the performance of Combined, SPADE and AllShowers on the GettingSquare dataset.

### *4.1* GettingHigh

Fig. 4 shows the result of training OmniJet-$\alpha_C$, Combined, SPADE and AllShowers on the GettingHigh dataset, compared to the Geant4 ground truth. Note that AllShowers predicts the per-layer hit count separately via PointCountFM before generation, giving it a structural advantage on panels (a) and (e) that the autoregressive models do not share.

The top row of Fig. 4 shows shower-level observables. For the total number of hits per shower, Fig. 4 (a), AllShowers reproduces the Geant4 distribution most accurately across the full range, largely inheriting this fidelity from its PointCountFM module. In contrast, the three autoregressive models, must determine the sequence length as part of the generation process itself. OmniJet-$\alpha_C$ samples a dedicated stop token as part of its vocabulary, while Combined and SPADE rely on the stop head described in Section 3.5. OmniJet-$\alpha_C$ shows the largest deviations, consistent with the fact that the stop decision is entangled with the sampling of physical tokens. Both Combined and SPADE describe the bulk of the distribution well, but deviate more strongly in the tail beyond roughly 1300 hits, suggesting that that the stop head may terminate long showers too late.

The ratio of deposited to incident energy, Fig. 4 (b), is captured best by SPADE, which shows fluctuations only in the tails. Note that this strong agreement relies on the post-processing step described in Appendix B, which actively filters out the 0.35% of generated SPADE showers that suffer from the early-stopping artifact and would otherwise populate an unphysical low-energy tail. Combined and AllShowers fluctuate across the full distribution, including the bulk. OmniJet-$\alpha_C$ is not conditioned on the incident energy in its current form and is therefore omitted from this panel, although such conditioning could in principle be added.

Spade also best reproduces the center of gravity along $x$, Fig. 4 (c). Because this metric is an energy-weighted spatial average, it relies heavily on the intra-hit correlation between $x_i$ and $E_i$, features that are separated by two delay steps in the SPADE input encoding. The accurately reproduced distribution therefore validates the design of the staggered conditioning scheme, indicating that the autoregressive structure recovers the energy–position correlation successfully.

The bottom row of Fig. 4 shows hit-level observables. The hit-energy spectrum, Fig. 4 (d), shows that SPADE provides the best agreement with Geant4 across three orders of magnitude, followed closely by Combined. Both models predict the hit energy through a dedicated continuous head conditioned on all three spatial coordinates of the same hit, allowing them to model the $E$ spectrum directly. In contrast, OmniJet-$\alpha_C$ encodes the hit energy jointly with the spatial coordinates through the VQ-VAE codebook, so the achievable energy resolution is bounded by the codebook granularity and visibly degrades in the tails of the spectrum. AllShowers deviates

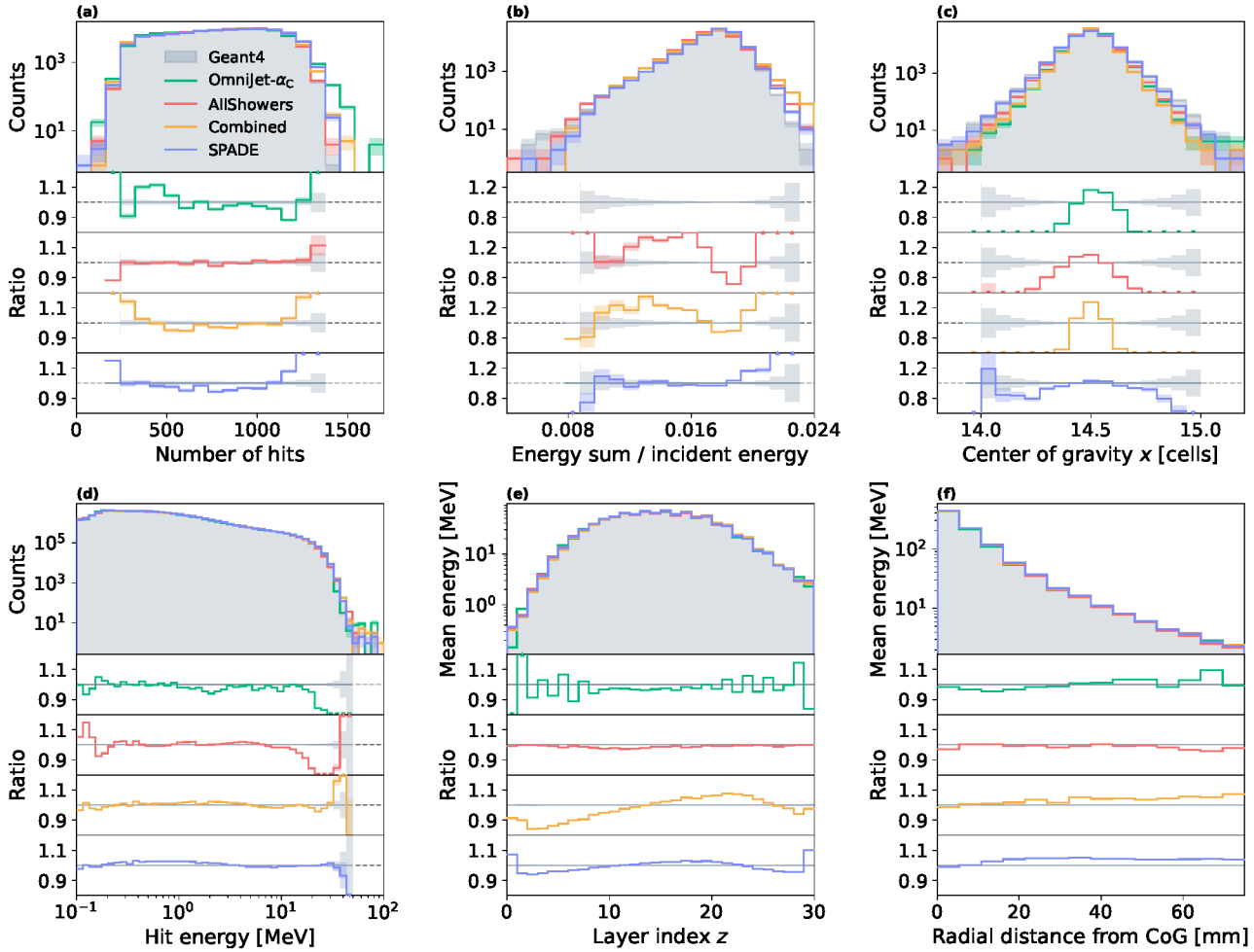



**Figure 4.** Comparison of OmniJet-$\alpha_C$, AllShowers, Combined and SPADE on GettingHigh photon showers uniformly distributed between 10 and 100 GeV. For each generator, 95k samples are shown. The shaded band represents the statistical standard deviation.

from Geant4 to a similar degree on this observable, which the authors of AllShowers attribute to limited training statistics in the sparsely populated tails of the spectrum [52].

The mean energy as a function of layer index, Fig. 4 (e), is best reproduced by AllShowers. This is to be expected, because of the PointCountFM (see Section 3.8). Among the autoregressive generators, SPADE outperforms both Combined and OmniJet-$\alpha_C$. OmniJet-$\alpha_C$ in particular shows a pronounced layer-by-layer bumpiness, consistent with the loss of fine-grained spatial information through the VQ-VAE tokenization.

The mean energy as a function of radial distance from the center of gravity, Fig. 4 (f), is best reproduced by AllShowers with some deviation at large radii. SPADE and Combined slightly overestimate the mean hit energy across the full radial range. OmniJet-$\alpha_C$ displays strong bin-to-bin fluctuations. We note that the distribution contains a contribution from the binning of the histogram itself, since the radial distance is a derived quantity computed from discrete sensor positions on the irregular ECAL grid.

### *4.2* GettingSquare

When evaluating models on GettingSquare, each model is trained at the granularity that maximizes its own fidelity at the evaluation resolution. AllShowers is trained at x16 with downstream remapping, Combined and SPADE directly at the evaluation grid (x1-merged). This ensures the comparison reflects each architecture at its best rather than a single imposed protocol; see Appendix C for the supporting evidence and distributions for the remaining granularities. As in the previous subsection, AllShowers serves primarily as an external reference, and the focus here is on whether the factorized SPADE embedding offers an advantage over the joint Combined embedding at the same effective granularity.

The top row of Fig. 5 shows shower-level observables. For the total number of hits per shower, Fig. 5 (a), all three models follow the Geant4 distribution within a few percent across the bulk of the range, with only minor

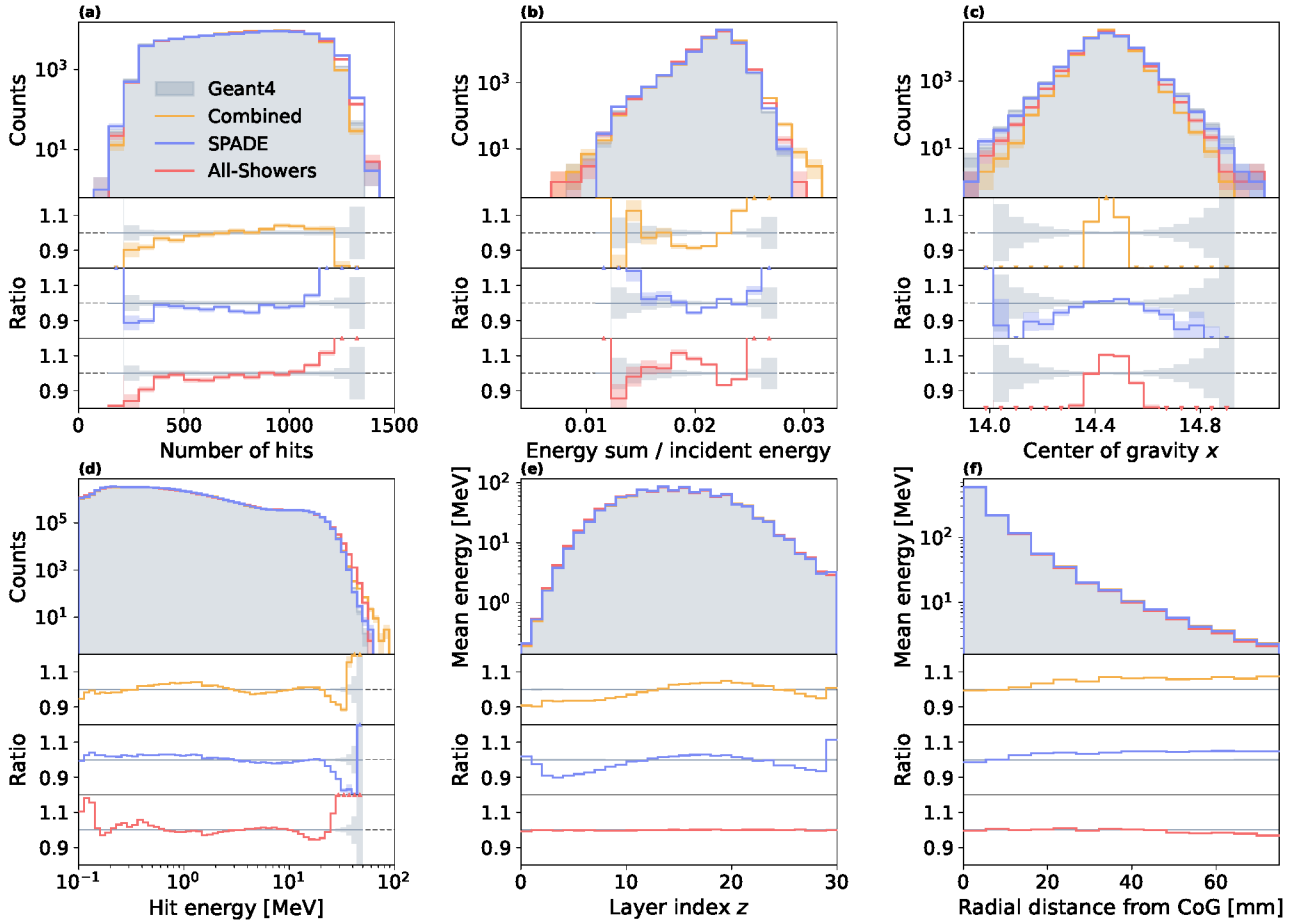



**Figure 5.** Comparison of AllShowers, Combined and SPADE on GettingSquare photon showers uniformly distributed between 10 and 100 GeV. For each generator, 95k samples are shown. AllShowers was trained on x16 granularity and then remapped to x1-merged, whereas Combined and SPADE were trained on x1-merged directly.

deviations in the high-multiplicity tail. The performance on the ratio of deposited to incident energy, Fig. 5 (b), is similar for all three models, with comparable levels of fluctuation around the truth. SPADE is arguably best in the bulk of the distribution.

The center of gravity along $x$, Fig. 5 (c), shows a clear separation between the two autoregressive models: SPADE follows the Geant4 distribution closely across the central peak, while Combined exhibits noticeably larger deviations in the ratio panel. As already discussed for GettingHigh, this observable is an energy-weighted spatial average and therefore directly sensitive to the intra-hit $(x, E)$ correlation. The SPADE advantage here is consistent with the factorized embedding distributing the training signal more uniformly across the spatial vocabulary: with $N_x + N_y + N_z$ rather than $N_x \cdot N_y \cdot N_z$ unique spatial vectors, every embedding is updated frequently, and the energy–position correlation is recovered without the token sparsity that affects Combined. AllShowers produces a slightly narrower distribution than Geant4, visible as an excess at the peak and a deficit in the tails of the ratio panel; SPADE matches Geant4 more closely than AllShowers across the full region.

The bottom row of Fig. 5 shows hit-level observables. The hit-energy spectrum, Fig. 5 (d), is reproduced with comparable fidelity by all three models across nearly three orders of magnitude, with only minor disagreements in the sparsely populated high-energy tail. The mean energy as a function of layer index, Fig. 5 (e), is most accurately reproduced by AllShowers due to its PointCountFM module, while SPADE and Combined behave essentially equivalently, both showing a mild dip relative to Geant4 in the middle layers. For the mean energy as a function of radial distance from the center of gravity, Fig. 5 (f), AllShowers reproduces the Geant4 distribution well across the full radial range. Combined develops a systematic upward drift toward large radii, overshooting Geant4 by up to roughly 10% at the outermost bins, while SPADE shows a slightly more moderate version of the same trend.

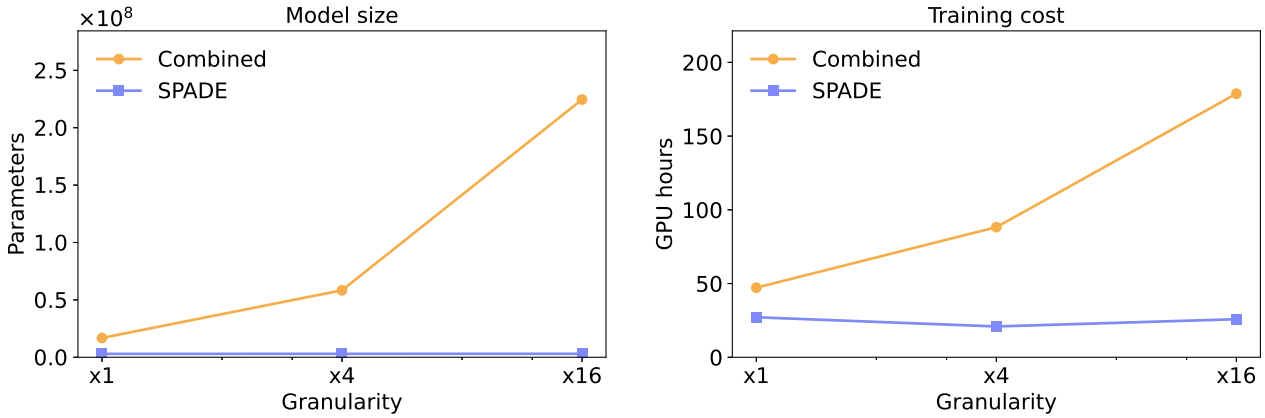



**Figure 6.** Left: the number of model parameters as a function of the granularity for SPADE and Combined. Right: the number of GPU hours needed to train each model within 2% of the the minimal loss reached after 55k steps.

## 5 Training efficiency

Due to the difference in tokenization strategy, the Combined model is substantially larger than SPADE. As shown in Section 3.2, Combined's spatial embedding layer and token-prediction head scale as $N^3$ with the granularity $N$, while SPADE's factorized counterparts scale as $3N$. The resulting parameter counts at each granularity are shown in Fig. 6 (left) and tabulated in Table 1, which additionally lists the granularity-independent transformer backbone shared by both models.

To compare wall-clock training cost on a fair basis across granularities and architectures, we define it as the number of GPU hours (NVIDIA H100) needed for each model to first reach within 2% of its own minimum validation loss within 55k training steps. The threshold is per-model rather than absolute, because SPADE and Combined converge to noticeably different minima at finer granularities. The 2% level corresponds to essentially-converged training, beyond which further steps yield diminishing returns. The resulting cost per model is shown in Fig. 6 (right) and tabulated in Table 1: at GettingSquare-x16, Combined requires 178.7 GPU hours against SPADE's 25.8 GPU hours, a factor of 6.9 more.

Two effects compound to produce this gap. First, Combined's large parameter count limits the number of training showers that fit on a single GPU, forcing the use of batch accumulation to reach the desired effective batch size. These extra accumulation steps translate directly into wall-clock cost. Second, SPADE needs fewer gradient steps to converge, a representational advantage of the factorized embedding. By dividing the output space across three smaller vocabularies instead of one large one, SPADE mitigates gradient sparsity and updates every token embedding more frequently.

The validation-loss trajectories underlying these numbers, including the markers at which each run first crosses the 2% threshold, are shown in Appendix D . The same curves also reveal that SPADE converges to a lower minimum validation loss than Combined at finer granularities.

**Table 1.** Model size and training cost vs. granularity. Combined's and SPADE's transformer block is granularity-independent; the remaining parameters come from granularity-dependent input/output heads At x16, SPADE uses a factor of 74 fewer parameters and trains 6.9 times faster than Combined.

| | Parameters | | | GPU hours | | |
|---|---|---|---|---|---|---|
| Model | x1 | x4 | x16 | x1 | x4 | x16 |
| Combined | 16.80 M | 58.35 M | 224.56 M | 47.2 | 88.2 | 178.7 |
| SPADE | 2.96 M | 2.98 M | 3.02 M | 27.1 | 20.9 | 25.8 |
| of which: transformer blocks | 2.81 M | 2.81 M | 2.81 M | — | — | — |

## 6 Inference time

We measure the wall-clock time to generate a single shower for Combined, SPADE, and AllShowers on a single NVIDIA A100-SXM4-80GB, sweeping incident energies (10, 55, 100 GeV) at batch size 4096 — the largest batch size that fits on a single GPU for all three models. We discard one warmup batch per setting and average over $N = 10$ independent batches, reporting mean $\pm$ standard error of the mean. The reported per-shower time includes autoregressive token generation and the mapping of tokens onto the GettingSquare-x1-merged geometry. To ensure a comparison at their respective highest fidelities, AllShowers is generating showers in the GettingSquare-x16 geometry, while SPADE and Combined are generating coarser showers in the

GettingSquare-x1-merged representation. The postprocessing step described in Appendix B is not included in the inference time for SPADE, as it only effects 0.35% of showers. The results are summarized in Table 2.

AllShowers is consistently the fastest model, generating showers roughly three times faster than SPADE and up to five times faster than Combined. This speedup occurs because AllShowers is not an autoregressive model generating individual hits sequentially; instead, it iterates over a small, fixed number of flow-matching integration steps. However, its generation time still grows with incident energy, because the computational cost of each network pass increases with the overall size of the generated point cloud.

In contrast, SPADE and Combined perform one cached transformer step per sequence position, with the KV cache growing linearly with the number of generated hits. Their per-shower time therefore scales directly with the average hit count, driving the increase in generation time at higher incident energies. Between the two autoregressive models, SPADE generates roughly twice as fast as Combined at both 55 and 100 GeV. This is consistent with the parameter scaling detailed in Table 1: Combined's much larger embedding and output heads introduce a per-step cost that compounds over the long autoregressive sequences.

**Table 2.** Per-shower inference time in ms/shower on a single NVIDIA A100-SXM4-80GB at batch size 4096, as a function of incident energy and model. Values are mean ± SEM in ms/shower across $N = 10$ independent batches.

| $E_{\text{inc}}$ [GeV] | SPADE | Combined | AllShowers |
|---|---|---|---|
| 10 | $12.72 \pm 0.05$ | $15.60 \pm 0.16$ | $3.71 \pm 0.05$ |
| 55 | $21.71 \pm 0.04$ | $46.15 \pm 0.17$ | $7.06 \pm 0.66$ |
| 100 | $33.44 \pm 0.12$ | $62.99 \pm 0.01$ | $11.90 \pm 0.01$ |

## 7 Conclusion

As autoregressive transformer models are increasingly being used outside of the language domain for which they were originally developed, it is important to revisit the original architectural strategies. Tokenization has been the go-to method to turn text into numbers that the model can ingest, but in domains like particle physics, where our input already is numerical, this might not be the ideal choice. Furthermore, physics data often lives in high-dimensional spaces, parts of which can be discrete. In this work we have explored how to deal with such a high-dimensional space without being bottlenecked by giant codebooks, as well as how to preserve the important correlations between the different features.

SPADE is applied to the crucial problem of simulating highly granular calorimeter data within a feasible compute budget. The introduction of the split head approach of SPADE enables the model to directly make use of the discrete features as-is. Compared to the original OmniJet-$\alpha_C$ model, it completely eliminates the need to train a VQ-VAE for tokenization. The feature correlations are preserved through the delay mechanism, where the different features become conditioned on one another. In comparison to the Combined model, which combines the spatial coordinates into a single number, SPADE enables easy scaling to higher granularities. In contrast, a higher granularity for the Combined model leads to a drastic inflation of the number of model parameters, which can make training and inference prohibitively expensive. We have shown that SPADE is easier and more efficient to train than both OmniJet-$\alpha_C$ and Combined, and that these technical improvements introduced here allow an autoregressive model for the first time to achieve a performance on par with state of the art models in generating realistic electromagnetic showers, without the penalty of a large Combined codebook.

The task of additionally simulating hadronic showers, which is a challenge made harder by the higher complexity of the data, is left for future work. Another avenue for future work relates to the inference time. Despite improving compared to the Combined model, SPADE is still slower than state of the art models like AllShowers due to the inherent sequential nature of autoregressive generation. However, speed is not the main point of SPADE. Rather it is the improvements over single-stream combined tokenization models, which, given the increased importance of autoregressive transformers for foundation models, is highly significant for models across a wide range of domains.

Beyond calorimeter simulation, the SPADE mechanism (split-and-delay embeddings) applies to any sequential generative task whose tokens carry multiple features, whether discrete or continuous. By embedding each feature independently and recovering their correlations through staggered conditioning, SPADE extends the next token prediction paradigm of large language models to multi-dimensional data, without lossy intermediate representations. This positions SPADE as a candidate building block for LLM-style pretraining-and-fine-tuning workflows in HEP and beyond, in which a single autoregressive backbone, pretrained on large unlabeled datasets of multi-feature tokens, is subsequently adapted to new detectors, particle species, or different problem domains with multi-dimensional sensor data. An immediate application could be in high-dimensional astrophysics data, where for example data from the Square-Kilometer-Array has already been used to train autoregressive transformer models [66].

## A Model details and hyperparameters

Model hyperparameters can be found in Table 3. The standardization statistics $\mu$ and $\sigma$ are computed once on the training set and applied to validation and test data unchanged.

**Table 3.** Hyperparameters for the SPADE and Combined models. The two models differ only in their embedding and unembedding scheme (top), while the transformer backbone, training, and sampling configuration are identical (bottom).

| Type | Parameter | SPADE | Combined |
|---|---|---|---|
| Embedding / Unembedding | input representation | per-axis $(x, y, z)$ | joint voxel index |
| | vocabulary size † | $N_x + N_y + N_z$ | $N_x N_y N_z$ |
| | at GettingHigh/GettingSquare-x1 | 96 | 27 002 |
| | at GettingSquare-x4 | 156 | 108 002 |
| | at GettingSquare-x16 | 276 | 432 002 |
| | output heads | three softmaxes | single softmax |

† The base vocabulary size is determined by the granularity. An additional two tokens (+2) account for the padding and missing-ID embeddings (see Eqs. (2) and (3)). These are added once globally for the Combined model, and per axis for SPADE (resulting in +6).

| Type | Parameter | Value |
|---|---|---|
| Data Preprocessing | hit energy | $\log E$, then $(x - \mu)/\sigma$ |
| | incident energy | $(x - \mu)/\sigma$ |
| | hit ordering | layer-ascending, energy-descending |
| Backbone | embedding dim | 256 |
| | layers | 3 |
| | attention heads | 8 |
| | MLP ratio | 4 |
| | positional encoding | RoPE |
| | attention | multi-query, FlashAttention |
| | dropout | 0 |
| | max sequence length | 1700 for GettingHigh and 4600 for GettingSquare |
| Energy head (MoG) | mixture components $K$ | 4 |
| | MLP layers | 2 |
| | MLP hidden dim | 128 |
| | relative energy noise $\sigma_{\text{noise}}$ | 0.05 |
| Stop head | spatial embedding dim | 16 |
| | energy projection dim | 16 |
| | position projection dim | 16 |
| | hidden layers | 2 |
| | hidden dim | 128 |
| | dropout | 0.2 |
| | BCE positive class weight | 1000 |
| Training | optimizer | Ranger (Lookahead + RAdam) |
| | Adam $\beta_1, \beta_2$ | 0.95, 0.999 |
| | Lookahead $\alpha, k$ | 0.25, 16 |
| | learning rate | $1 \times 10^{-4}$ (constant after warmup) |
| | warmup steps | 2 000 |
| | weight decay | $5 \times 10^{-3}$ |
| | per-GPU batch size | 48 |
| | gradient clipping | 1.0 |
| Inference | weights | EMA, decay 0.999 |
| | sampling temperature | 1.0 |
| | stop threshold | $p > 0.98$ |

## B Postprocessing

Autoregressive models can accumulate errors step by step, and once the sampled shower drifts too far from the training distribution, the model can *hallucinate* showers that no longer resemble physically valid ones. In our setting the dominant failure mode is the total deposited energy $E_{\text{sum}} = \sum_i E_i$ falling far below the range observed in Geant4 at the same incident energy. To mitigate this we fit a lower envelope to the Geant4 showers $E_{\text{sum}}$ vs. $E_{\text{inc}}$ distribution and reject generated samples that fall below it. A shower is flagged if:

$$E_{\text{sum}} \;<\; s \cdot \left(a\, E_{\text{inc}}^{b} + c\right), \tag{5}$$

where $(a, b, c)$ are fit to a low percentile of the $E_{\text{sum}}$ vs. $E_{\text{inc}}$ relation in Geant4, and the safety factor $s = 0.9$ pushes the threshold further below the fitted envelope. For the GettingSquare datasets we use $a = 0.0174$, $b = 0.9379$, $c = -0.0180$ (with $E_{\text{inc}}$ and $E_{\text{sum}}$ in GeV), fit to the lower envelope of the Geant4 $E_{\text{sum}}$–$E_{\text{inc}}$ relation. Flagged showers are regenerated under the same conditioning and replace the original in place; we allow up to $k_{\max} = 2$ regeneration rounds, after which any still-flagged shower is kept unchanged. Applying the same threshold to the GettingSquare test set flags $f_{\text{G4}} = 0.014\%$ of real showers, with the rate roughly flat across incident energy bins (between 0.006% and 0.026% across the 10–90 GeV range).

On generated GettingSquare showers, the first-pass flag rate is $f_{\text{SPADE}} = 0.35\% \pm 0.03\%$ (Poisson on $\sim$140/40,000 counts) for SPADE and $f_{\text{Combined}} = 0.015\% \pm 0.006\%$ (Poisson on $\sim$6/40,000 counts) for Combined, both falling to 0% after $k_{\max} = 2$ regeneration rounds. We treat the residual bias as a known systematic of the procedure. Combined's flag rate is statistically indistinguishable from the Geant4 baseline, indicating that this failure mode is essentially absent for Combined and specific to SPADE. The cause of this difference between the two encodings is not yet understood, but the postprocessing filter removes the artifact for SPADE.

## C Generation quality across granularities

Section 4.2 focused on the GettingSquare-x1-merged representation, the common grid on which all three models can be compared directly. Here we provide the full set of distributions for the remaining GettingSquare variants. For each transverse granularity we show two figures: the models trained and evaluated at their *native* resolution (Figs. 7, 9 and 11), and the same generated showers after a post-sampling remapping onto the x1-merged grid (Figs. 8, 10 and 12). For the x4 and x16 granularities, the remapping re-bins the higher-resolution output down to $30 \times 30 \times 30$ and merges hits that share a voxel. For x1, the grid is unchanged and only the merging step applies.

The central observation across these six configurations is that Combined and SPADE retain their performance as the native granularity increases: at x1, x4 and x16 (Figs. 7, 9 and 11) both models reproduce the Geant4 reference well. Remapping a model trained at a higher granularity down to the x1-merged grid, however, does not improve the agreement, and the two encodings respond to it very differently. SPADE stays close to the Geant4 reference across the remapped configurations, but Combined degrades visibly, most clearly in the hit multiplicity (Fig. 8(a), Fig. 10(a) and Fig. 12(a)): merging onto a coarser voxel grid barely changes its hit count, meaning too few of its generated hits coincide on the same voxel to be collapsed by the merge. Combined therefore samples spatial positions too diffusely across the vocabulary, missing the dense local clusters that characterize real showers. The factorized SPADE embedding, by contrast, learns these spatial correlations sharply enough that its merged hit count matches Geant4 — evidence that the factorization helps SPADE capture the actual hit geometry, not merely the aggregate distributions.

These remapping results also explain a methodological choice in the main comparison. In Section 4.2 each model is evaluated under the training strategy that maximizes its own fidelity, so that the comparison reflects each architecture at its best rather than a single imposed protocol. The present results show what that strategy is for the autoregressive models: training at a higher granularity and remapping down brings no gain for SPADE and breaks the hit multiplicity for Combined, so both are trained directly on x1-merged. AllShowers sits at the opposite end of this trade-off. As noted in Section 4.2, it is trained at x16 and remapped to x1-merged, because training it directly at the lowest granularity produces artifacts (Fig. 7(f) and Fig. 8(f)). This is consistent with the larger literature, where diffusion and flow-matching-based generators are generally more accurate when trained at the highest available granularity and mapped back to the detector space [67]. SPADE is the exception: it is most accurate trained directly at the evaluation granularity, with no high-resolution intermediate stage at all.

One caveat remains. Of the native datasets used here, only x16 is free of duplicate hits sharing a voxel; the x1 and x4 variants retain duplicates that no physically realizable readout could record. A real detector at those granularities would therefore measure a lower hit multiplicity per shower than the corresponding training data. The remapped variants collapse these duplicates and are physical in this sense. Whether the granularity behavior reported here generalizes to other detector geometries remains to be evaluated. Even so, the comparison provides a useful first characterization of how the factorized (SPADE) and joint (Combined) encodings respond to changes in detector segmentation.

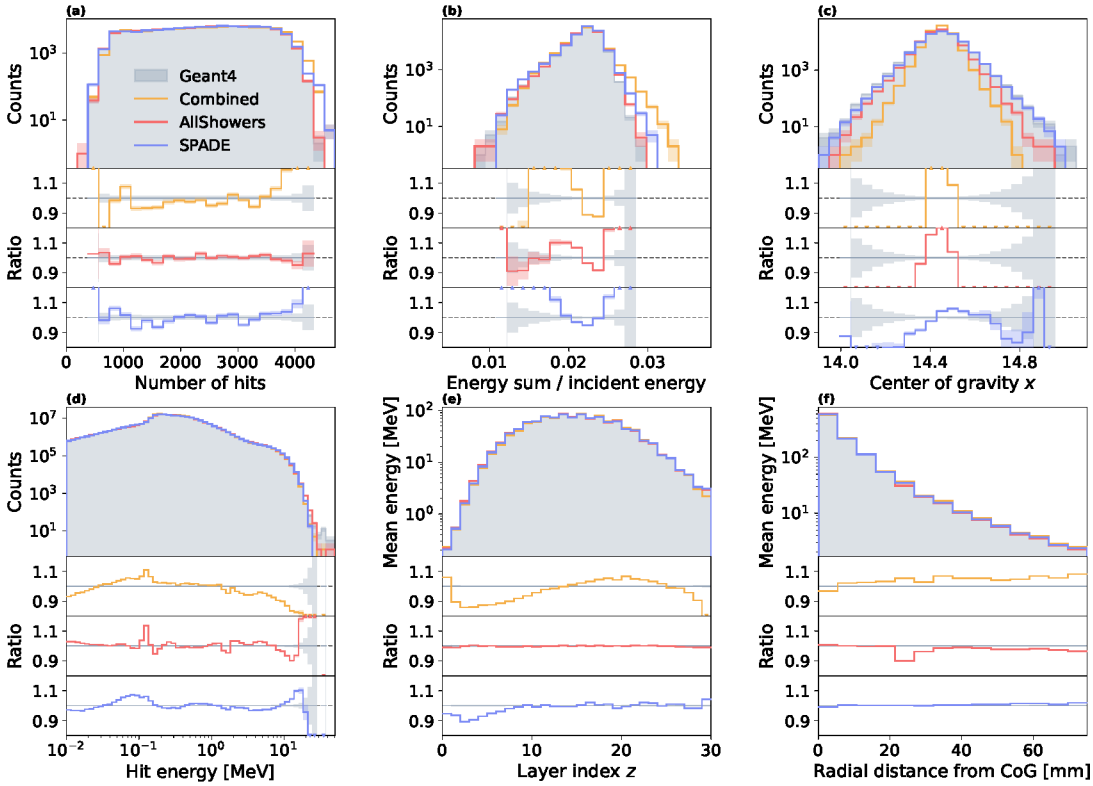



**Figure 7.** Trained on GettingSquare-x1 at native resolution ($30 \times 30 \times 30$). Comparison of AllShowers, Combined and SPADE on photon showers uniformly distributed between 10 and 100 GeV. For each generator, 95k samples are shown; the shaded band is the statistical standard deviation.

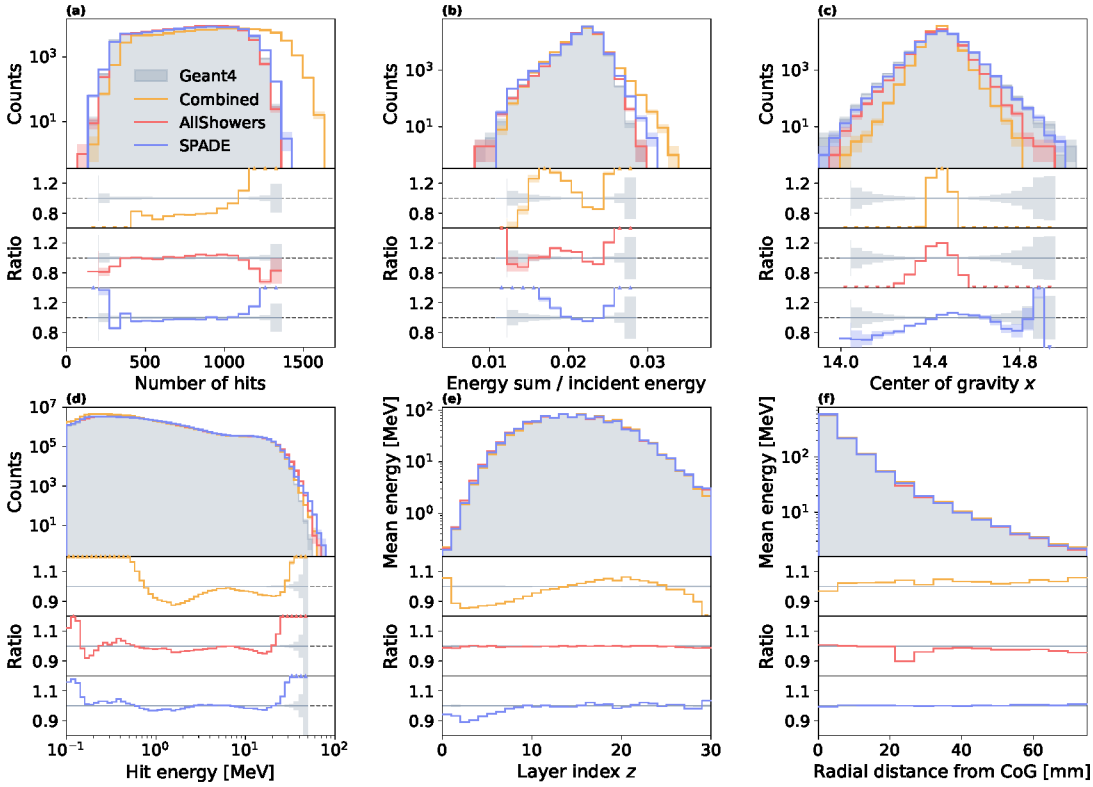



**Figure 8.** Models trained on GettingSquare-x1 with generated showers merged to the x1-merged grid. Otherwise as Fig. 7.

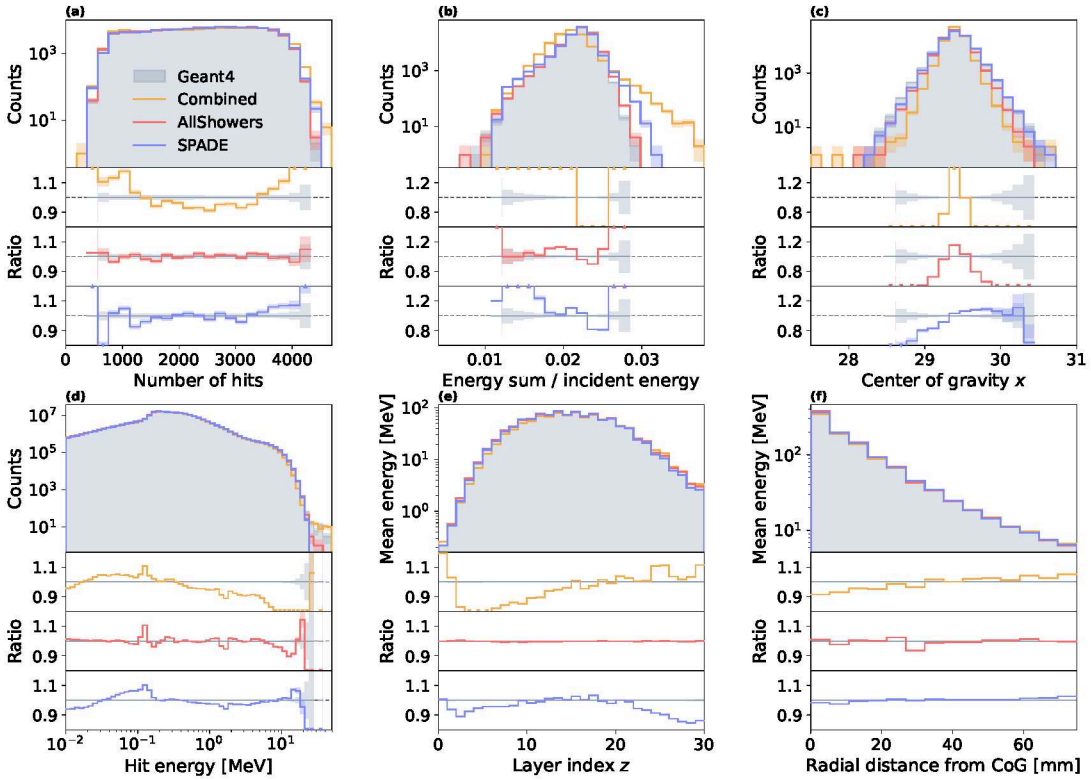



**Figure 9.** Trained on GettingSquare-x4 at native resolution ($60 \times 60 \times 30$). Comparison of AllShowers, Combined and SPADE on photon showers uniformly distributed between 10 and 100 GeV. For each generator, 95k samples are shown; the shaded band is the statistical standard deviation.

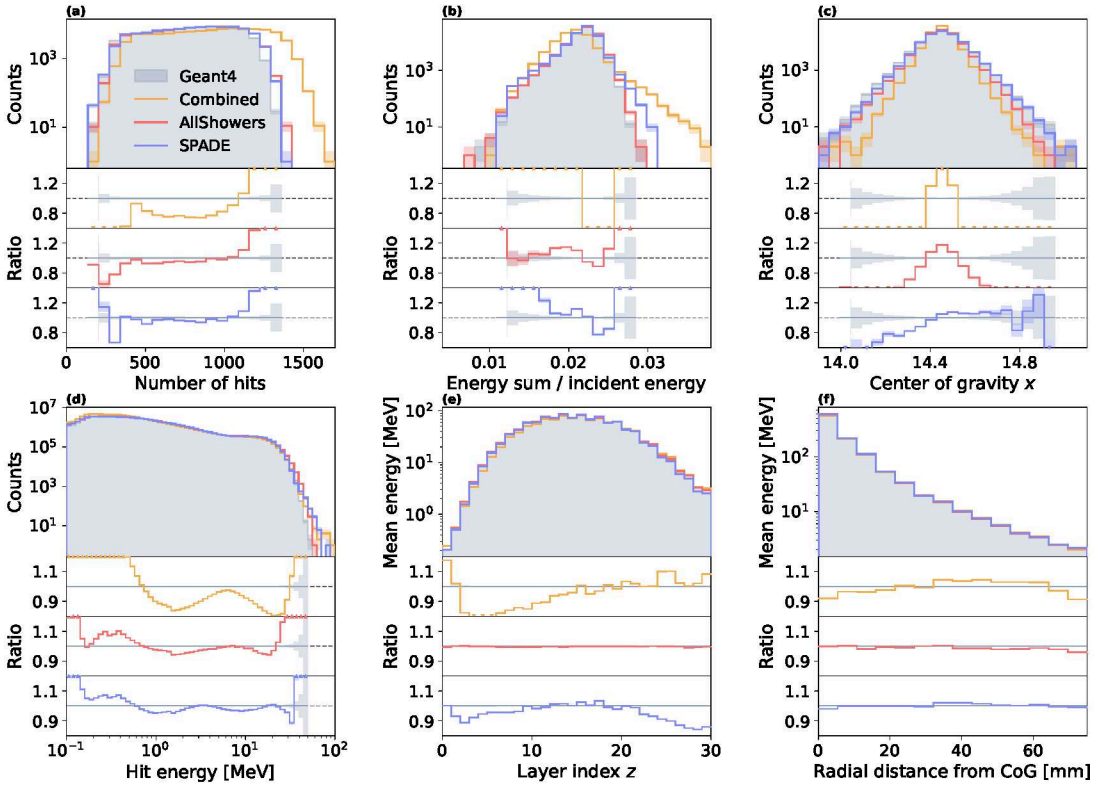



**Figure 10.** Models trained on GettingSquare-x4 with generated showers remapped to the x1-merged grid. Otherwise as Fig. 9.

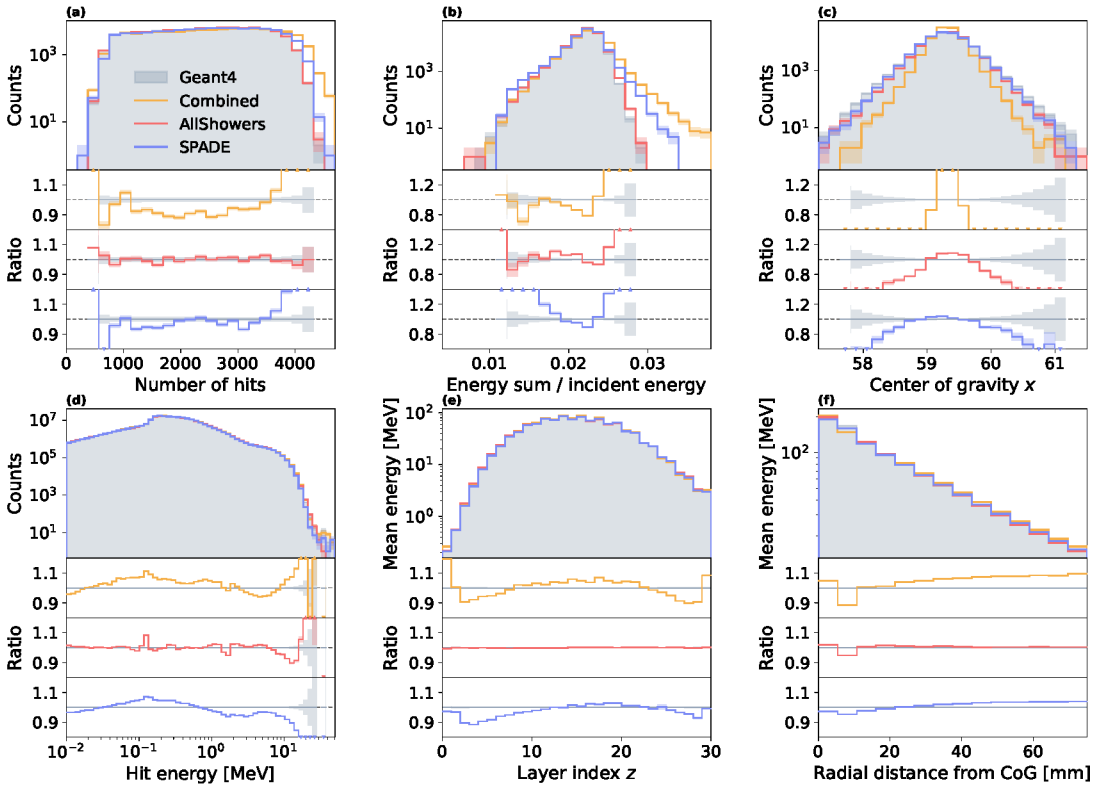



**Figure 11.** Trained on GettingSquare-x16 at native resolution ($120 \times 120 \times 30$). Comparison of AllShowers, Combined and SPADE on photon showers uniformly distributed between 10 and 100 GeV. For each generator, 95k samples are shown; the shaded band is the statistical standard deviation.

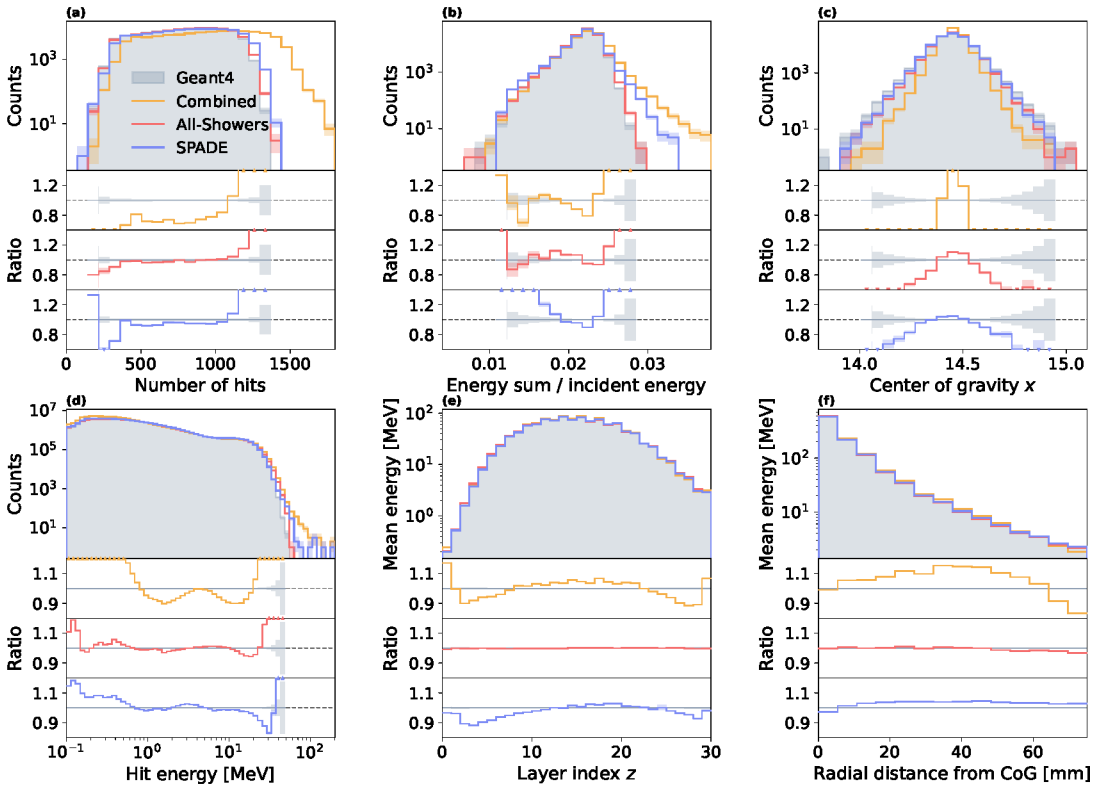



**Figure 12.** Models trained on GettingSquare-x16 with generated showers remapped to the x1-merged grid. Otherwise as Fig. 11.

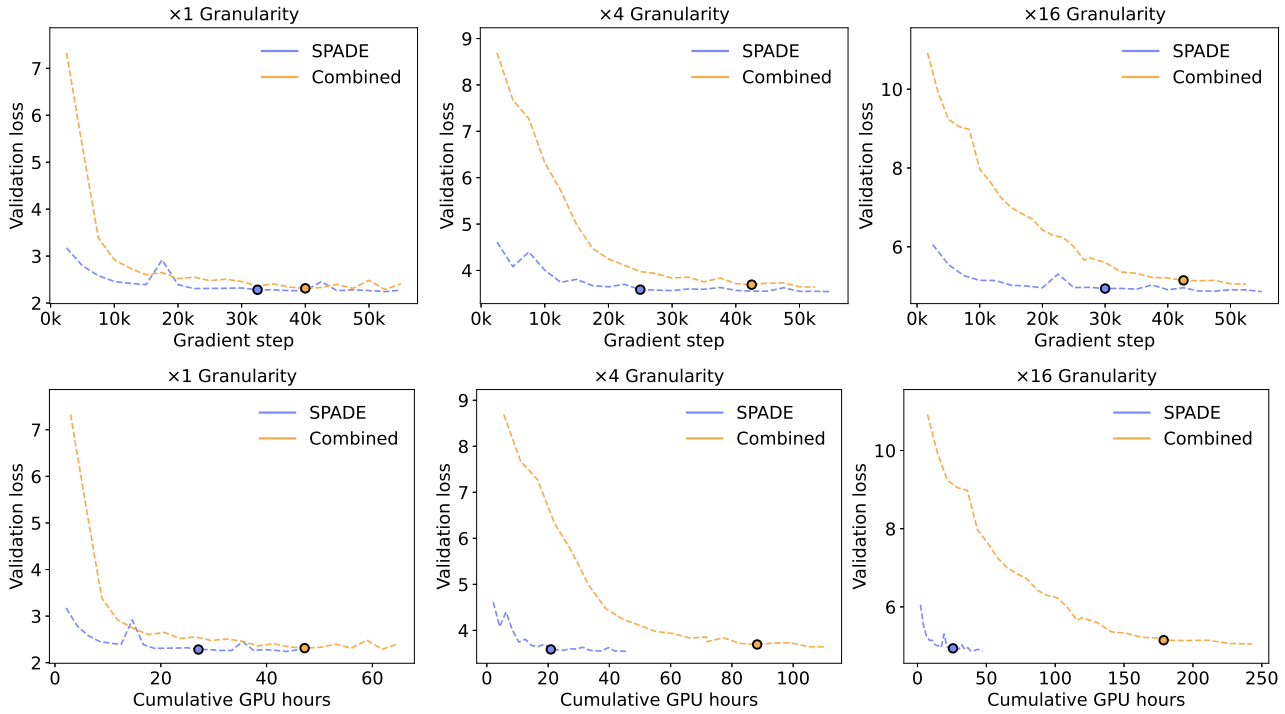



**Figure 13.** Validation loss for SPADE and Combined at the three GettingSquare granularities, versus gradient step (top) and cumulative GPU hours on an NVIDIA H100 (bottom). Markers: first point at which each run reaches within 2% of its own minimum. SPADE matches or undercuts Combined's minimum validation loss and reaches it at a much lower GPU-hour cost at every granularity, with the gap widening with finer segmentation.

## D Convergence

Fig. 13 shows the validation-loss trajectories underlying the training-efficiency summary of Section 5, plotted against the gradient step (top) and cumulative GPU hours (bottom). The markers indicate the first point at which each run reaches within 2% of its own minimum, the threshold used in Table 1 and Fig. 6.

The trajectories make two features visible that the aggregate numbers do not capture. First, SPADE descends more steeply than Combined from the earliest steps. This is the visual signature of the reduced gradient sparsity discussed in Section 5. Second, the gap between the two dotted reference lines widens with granularity: at x1 the two minima are essentially tied, while at x16 SPADE settles a clear margin below Combined. The factorized embedding therefore yields not only a faster route to convergence but also a lower endpoint at higher detector granularity.

## Code Availability

The code for this paper can be found at https://github.com/uhh-pd-ml/SPADE.

## Data Availability

The code for recreating the dataset can be found at https://github.com/FLC-QU-hep/getting_high.


## Acknowledgments

We thank Sarah Heim for helpful comments on the manuscript. We thank Thorsten Buss for valuable discussions on FlashAttention and training strategies, and for his help in providing additional scripts to adapt the AllShowers model to the new datasets. We thank Anatolii Korol for providing the data used to create the GettingSquare dataset.

## Funding

J.B., A.H., G.K., M.M. and H.R. are supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the German Excellence Initiative – EXC 2121 Quantum Universe – 390833306, by PUNCH4NFDI – project number 460248186, as well as via the SciFM consortium (05D25GU4) funded by the German Federal Ministry of Research, Technology, and Space (BMFTR) in the ErUM-Data action plan. JB also acknowledges support via the Hamburg VISTA/VISOR — Virtual Initiative for Science & Technology in AI

— network. This work has used the the Maxwell computational resources at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany, as well as the HPC-cluster Hummel-2 at University of Hamburg. Hummel-2 was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 498394658.

## References

[1] Agostinelli S *et al.* (GEANT4) 2003 *Nucl. Instrum. Meth.* **A506** 250–303

[2] Albrecht J *et al.* 2019 *Computing and Software for Big Science* **3** ISSN 2510-2044 URL http://dx.doi.org/10.1007/s41781-018-0018-8

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

[4] Paganini M, de Oliveira L and Nachman B 2018 *Phys. Rev. Lett.* **120** 042003 (*Preprint* https://arxiv.org/abs/1705.02355)

[5] Paganini M, de Oliveira L and Nachman B 2018 *Phys. Rev. D* **97** 014021 (*Preprint* https://arxiv.org/abs/1712.10321)

[6] de Oliveira L, Paganini M and Nachman B 2018 *J. Phys. Conf. Ser.* **1085** 042017 (*Preprint* https://arxiv.org/abs/1711.08813)

[7] Erdmann M, Geiger L, Glombitza J and Schmidt D 2018 *Comput. Softw. Big Sci.* **2** 4 (*Preprint* https://arxiv.org/abs/1802.03325)

[8] Erdmann M, Glombitza J and Quast T 2019 *Comput. Softw. Big Sci.* **3** 4 (*Preprint* https://arxiv.org/abs/1807.01954)

[9] Musella P and Pandolfi F 2018 *Comput. Softw. Big Sci.* **2** 8 (*Preprint* https://arxiv.org/abs/1805.00850)

[10] Belayneh D *et al.* 2020 *Eur. Phys. J. C* **80** 688 (*Preprint* https://arxiv.org/abs/1912.06794)

[11] Butter A, Diefenbacher S, Kasieczka G, Nachman B and Plehn T 2021 *SciPost Phys.* **10** 139 (*Preprint* https://arxiv.org/abs/2008.06545)

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

[13] ATLAS Collaboration 2020 *J. Phys. Conf. Ser.* **1525** 012077

[14] ATLAS Collaboration 2022 *Comput. Softw. Big Sci.* **6** 7 (*Preprint* https://arxiv.org/abs/2109.02551)

[15] ATLAS Collaboration 2024 *Comput. Softw. Big Sci.* **8** 7 (*Preprint* https://arxiv.org/abs/2210.06204)

[16] Faucci Giannelli M and Zhang R 2024 *Eur. Phys. J. Plus* **139** 597 (*Preprint* https://arxiv.org/abs/2309.06515)

[17] Simsek E, Isildak B, Dogru A, Aydogan R, Bayrak A B and Ertekin S 2024 *PTEP* **2024** 083C01 (*Preprint* https://arxiv.org/abs/2401.02248)

[18] Cresswell J C, Ross B L, Loaiza-Ganem G, Reyes-Gonzalez H, Letizia M and Caterini A L 2022-11 CaloMan: Fast generation of calorimeter showers with density estimation on learned manifolds *36th Conference on Neural Information Processing Systems: Workshop on Machine Learning and the Physical Sciences* (*Preprint* https://arxiv.org/abs/2211.15380)

[19] Hoque S, Jia H, Abhishek A, Fadaie M, Toledo-Marín J Q, Vale T, Melko R G, Swiatlowski M and Fedorko W T 2024 *Eur. Phys. J. C* **84** 1244 (*Preprint* https://arxiv.org/abs/2312.03179)

[20] Liu Q, Shimmin C, Liu X, Shlizerman E, Li S and Hsu S C 2024-05 Calo-VQ: Vector-Quantized Two-Stage Generative Model in Calorimeter Simulation (*Preprint* https://arxiv.org/abs/2405.06605)

[21] Krause C and Shih D 2023 *Phys. Rev. D* **107** 113003 (*Preprint* https://arxiv.org/abs/2106.05285)

[22] Krause C and Shih D 2023 *Phys. Rev. D* **107** 113004 (*Preprint* https://arxiv.org/abs/2110.11377)

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

[24] Krause C, Pang I and Shih D 2024 *SciPost Phys.* **16** 126 (*Preprint* https://arxiv.org/abs/2210.14245)

[25] Xu A, Han S, Ju X and Wang H 2024 *JINST* **19** P02003 (*Preprint* https://arxiv.org/abs/2303.10148)

[26] Buckley M R, Krause C, Pang I and Shih D 2024 *Phys. Rev. D* **109** 033006 (*Preprint* https://arxiv.org/abs/2305.11934)

[27] Pang I, Shih D and Raine J A 2024 *Phys. Rev. D* **109** 092009 (*Preprint* https://arxiv.org/abs/2308.11700)

[28] Ernst F, Favaro L, Krause C, Plehn T and Shih D 2025 *SciPost Phys.* **18** 081 (*Preprint* https://arxiv.org/abs/2312.09290)

[29] Schnake S, Krücker D and Borras K 2024-03 CaloPointFlow II Generating Calorimeter Showers as Point Clouds (*Preprint* https://arxiv.org/abs/2403.15782)

[30] Du H, Krause C, Mikuni V, Nachman B, Pang I and Shih D 2025 *Phys. Rev. D* **111** 076004 (*Preprint* https://arxiv.org/abs/2404.18992)

[31] Majerz E, Dzwinel W and Kitowski J 2025-12 Inverse Autoregressive Flows for Zero Degree Calorimeter fast simulation *39th Annual Conference on Neural Information Processing Systems: Includes Machine Learning and the Physical Sciences (ML4PS)* (*Preprint* https://arxiv.org/abs/2512.20346)

[32] Mikuni V and Nachman B 2022 *Phys. Rev. D* **106** 092009 (*Preprint* https://arxiv.org/abs/2206.11898)

[33] Acosta F T, Mikuni V, Nachman B, Arratia M, Karki B, Milton R, Karande P and Angerami A 2024 *JINST* **19** P05003 (*Preprint* https://arxiv.org/abs/2307.04780)

[34] Mikuni V and Nachman B 2024 *JINST* **19** P02001 (*Preprint* https://arxiv.org/abs/2308.03847)

[35] Amram O and Pedro K 2023 *Phys. Rev. D* **108** 072014 (*Preprint* https://arxiv.org/abs/2308.03876)

[36] Jiang C, Qian S and Qu H 2025 *SciPost Phys.* **18** 195 (*Preprint* https://arxiv.org/abs/2401.13162)

[37] Kobylianskii D, Soybelman N, Dreyer E and Gross E 2024 *Phys. Rev. D* **110** 072003 (*Preprint* https://arxiv.org/abs/2402.11575)

[38] Jiang C, Qian S and Qu H 2024-04 BUFF: Boosted Decision Tree based Ultra-Fast Flow matching (*Preprint* https://arxiv.org/abs/2404.18219)

[39] Favaro L, Ore A, Schweitzer S P and Plehn T 2025 *SciPost Phys.* **18** 088 (*Preprint* https://arxiv.org/abs/2405.09629)

[40] Hildebrandt R, Kourlitis E, Hashemi B, Bünstorf M, Meyer T, Boskov N, Kagan M, Rosenbaum D, Ganguly S and Heinrich L 2026 Bricks: Compositional neural markov kernels for zero-shot radiation-matter simulation (*Preprint* 2605.06591) URL https://arxiv.org/abs/2605.06591

[41] Li W, Shao D Y, Shi H Z and Sun Y X 2026 Nested-gpt for variable-multiplicity parton showers: A case study in the resummation of non-global logarithms (*Preprint* 2605.18360) URL https://arxiv.org/abs/2605.18360

[42] Bommasani R *et al.* 2022 On the opportunities and risks of foundation models (*Preprint* 2108.07258) URL https://arxiv.org/abs/2108.07258

[43] Hallin A 2025 Foundation models for high-energy physics *2nd European AI for Fundamental Physics Conference* (*Preprint* 2509.21434)

[44] Birk J, Hallin A and Kasieczka G 2024 *Mach. Learn. Sci. Tech.* **5** 035031 (*Preprint* 2403.05618)

[45] Radford A, Narasimhan K, Salimans T and Sutskever I 2018 Improving language understanding by generative pre-training URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[46] van den Oord A, Vinyals O and Kavukcuoglu K 2018 Neural discrete representation learning (*Preprint* 1711.00937)

[47] Bao H, Dong L, Piao S and Wei F 2022 BEiT: BERT Pre-Training of Image Transformers (*Preprint* 2106.08254)

[48] Huh M, Cheung B, Agrawal P and Isola P 2023 Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks (*Preprint* 2305.08842)

[49] Birk J, Gaede F, Hallin A, Kasieczka G, Mozzanica M and Rose H 2025 *JINST* **20** P07007 (*Preprint* 2501.05534)

[50] Cardona-Giraldo C, Fanelli C, Giroux J, Granger C, Nachman B and Sabin G 2026 Generalizable foundation models for calorimetry via mixtures-of-experts and parameter efficient fine tuning (*Preprint* 2603.28804) URL https://arxiv.org/abs/2603.28804

[51] Copet J, Kreuk F, Gat I, Remez T, Kant D, Synnaeve G, Adi Y and Défossez A 2024 Simple and controllable music generation (*Preprint* 2306.05284) URL https://arxiv.org/abs/2306.05284

[52] Buss T, Day-Hall H, Gaede F, Kasieczka G and Krüger K 2026 AllShowers: One model for all calorimeter showers (*Preprint* 2601.11716)

[53] Buhmann E, Diefenbacher S, Eren E, Gaede F, Kasieczka G, Korol A and Krüger K 2021 *Comput. Softw. Big Sci.* **5** 13 (*Preprint* 2005.05334)

[54] Frank M, Gaede F, Grefe C and Mato P 2014 *Journal of Physics: Conference Series* **513** 022010 URL https://doi.org/10.1088/1742-6596/513/2/022010

[55] Abramowicz H *et al.* (ILD Concept Group) 2020 International Large Detector: Interim Design Report (*Preprint* 2003.01116)

[56] Suehara T *et al.* 2018 *JINST* **13** C03015 (*Preprint* 1801.02024)

[57] Ba J L, Kiros J R and Hinton G E 2016 Layer normalization (*Preprint* 1607.06450) URL https://arxiv.org/abs/1607.06450

[58] Bishop C 1994 Mixture density networks WorkingPaper 4288 Aston University copyright © 1994, Christopher M. Bishop. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (https://creativecommons.org/licenses/by-nc-nd/4.0/).

[59] Su J, Ahmed M, Lu Y, Pan S, Bo W and Liu Y 2024 *Neurocomputing* **568** 127063 ISSN 0925-2312 URL https://www.sciencedirect.com/science/article/pii/S0925231223011864

[60] Shazeer N 2019 Fast transformer decoding: One writer, en heads (*Preprint* 1911.02150)

[61] Dao T, Fu D Y, Ermon S, Rudra A and Ré C 2022 Flashattention: Fast and memory-efficient exact attention with io-awareness *Advances in Neural Information Processing Systems* vol 35 pp 10488–10500

[62] Dao T 2024 Flashattention-2: Faster attention with better parallelism and work partitioning *The Twelfth International Conference on Learning Representations*

[63] Zhang M, Lucas J, Ba J and Hinton G E 2019 Lookahead optimizer: k steps forward, 1 step back *Advances in Neural Information Processing Systems* vol 32 ed Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E and Garnett R (Curran Associates, Inc.) URL https://proceedings.neurips.cc/paper_files/paper/2019/file/90fd4f88f588ae64038134f1eeaa023f-Paper.pdf

[64] Liu L, Jiang H, He P, Chen W, Liu X, Gao J and Han J 2020 On the variance of the adaptive learning rate and beyond *International Conference on Learning Representations* URL https://openreview.net/forum?id=rkgz2aEKDr

[65] Tarvainen A and Valpola H 2018 Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results (*Preprint* 1703.01780) URL https://arxiv.org/abs/1703.01780

[66] Heneka C, Nieser F, Ore A, Plehn T and Schiller D 2026 *SciPost Phys.* **20** 070 (*Preprint* 2506.14757)

[67] Buhmann E, Diefenbacher S, Eren E, Gaede F, Kasicezka G, Korol A, Korcari W, Krüger K and McKeown P 2023 *Journal of Instrumentation* **18** P11025 ISSN 1748-0221 URL http://dx.doi.org/10.1088/1748-0221/18/11/P11025