Transformers for Generalized Fast Shower Simulation

Piyush Raikwar[1*], *Renato* Cardoso[1], *Nadezda* Chernyavskaya[1], *Kristina* Jaruskova[1], *Witold* Pokorski[1], *Dalila* Salamani[1], *Mudhakar* Srivatsa[2], *Kalliopi* Tsolaki[1], *Sofia* Vallecorsa[1], and *Anna* Zaborowska[1]

[1]CERN, Geneva, Switzerland
[2]IBM T. J. Watson Research Center, Yorktown Heights, NY USA

Abstract. Recently, transformer-based foundation models have proven to be a generalized architecture applicable to various data modalities, ranging from text to audio and even a combination of multiple modalities. Transformers by design should accurately model the non-trivial structure of particle showers thanks to the absence of strong inductive bias, better modeling of long-range dependencies, and interpolation and extrapolation capabilities. In this paper, we explore a transformer-based generative model for detector-agnostic fast shower simulation, where the goal is to generate synthetic particle showers, i.e., the energy depositions in the calorimeter. When trained with an adequate amount and variety of showers, these models should learn better representations compared to other deep learning models, and hence should quickly adapt to new detectors. In this work, we will show the prototype of a transformer-based generative model for fast shower simulation, as well as explore certain aspects of transformer architecture such as input data representation, sequence formation, and the learning mechanism for our unconventional shower data.

1 Introduction

Particles traversing HEP detectors are simulated with Geant4 [1]. As a large consumer of CPU resources [2], it is a target of many studies with the goal to accelerate it. For many detectors, the particle shower simulation in calorimeters takes the most considerable fraction of simulation time, therefore most efforts focus on the fast shower simulation.

Besides classical parameterization-based fast simulation [3], machine learning-based fast simulation has emerged to be a promising approach following its success in other fields. There have been many works in this regard with various types of generative models to generate an “image” of particle shower given initial conditions such as the angle of incidence and the energy of the particle. For example, Geant4 example Par04 [4] uses a variational autoencoder (VAE) to generate the voxel energies. AtlFast3 [5] uses generative adversarial networks (GANs) to generate realistic-looking showers by learning the loss function using a discriminator. Following the success of diffusion models in generating high-quality samples whilst being easy to train compared to GANs, CaloDiffusion [6] shows its applicability for fast simulation. Further, to address the sparsity of voxelized showers, a point-cloud representation of the shower is combined with GANs [7] or diffusion [8]. There are many other prior

[*] e-mail: piyush.raikwar@cern.ch

works such as [9–12], which improves the fast simulation in terms of speed, accuracy, supporting more initial conditions, etc. However, one common aspect in all the above-mentioned methods is that once trained, these models are capable of generating showers for only one detector. Designing a model for each calorimeter requires dedicated expertise and is computationally expensive. Therefore, in this work, inspired by the foundation models [13] and MetaHEP [14], we aim to explore a transformer-based detector-agnostic model capable of generating showers (once adapted) for ideally any detector.

Due to the complexity of the transformer, we decided to first explore components like sequence formation and position information. So, instead of directly designing a generative model using transformers, we first investigate the architecture. Next, we build a generative model and train it on a single detector. Ultimately, the objective is to scale both the model and dataset and conduct large-scale training to obtain a generalizable model adaptable to new detectors. This paper focuses on architecture exploration and the results of a generative model attained on a single detector with a reduced energy range.

The paper is structured as follows. In Section 2, we explain foundation models and their relevance for fast simulation. Section 3 explains the dataset structure. We explore the know-how around the transformer architecture in Section 4 and discuss associated experiments. Then in Section 5, we present our prototype of the generative model for fast simulation and preliminary results. Finally, we conclude and discuss future work in Section 6.

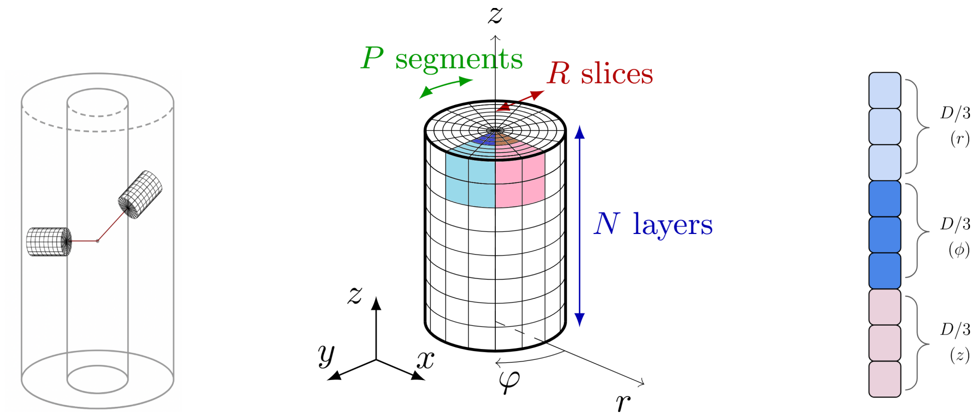
2 Foundation Models

The idea of foundation models started with large pre-trained language models like BERT [15] and GPT-3 [16]. Nowadays, they are not only limited to generating text. For example, Dall-E-2 [17], Imagen [18] are text-to-image models, GATO [19] is a multi-modal model capable of doing multiple tasks ranging from playing games to captioning an image, and many others. Most of these models share common attributes, including training typically on very large and diverse datasets coupled with a transformer-based [20] architecture, allowing them to learn good representations that help them in generalizing to a variety of different tasks.

The traits of foundation models are important to us in two key ways. First, a transformer is a generalized architecture without any inductive bias that can work with any type of data, e.g., text, image, audio, etc. as long as the input can be represented in terms of a sequence. The lack of strong inductive bias enables them to learn better representations given enough data. In our context, this translates to training the model on various detectors to learn a good representation of our shower data and then adapting the model to the desired detector. Second, the attention mechanism in transformers helps in modeling long-range dependencies. In our case, this translates to better learning the non-trivial structure of the shower and accurate modeling of faraway voxel energies within a shower.

3 Dataset

For training our model we utilize High Granularity Electromagnetic Calorimeter Shower Images [21] published on Zenodo. The dataset contains particle showers initiated by electrons represented as voxel energies in a cylindrical scoring mesh as the particle enters the calorimeter (Figure 1a). For each shower, the dataset also contains conditions such as the angle of incidence (θ), and the energy of the incident particle (E). Given the radial symmetry of the detector, we do not condition on the azimuthal angle, but that would be part of future work. In our case, to expedite the initial experiments, we begin by utilizing a subset of the dataset, specifically, we use a sampling calorimeter with silicon and tungsten layers with the angle



(a) (b) (c)

Figure 1: (a) Scoring meshes inside a detector. (b) Structure of the shower in r, ϕ, z directions. It further shows example patches with $\Delta r \times \Delta\phi \times \Delta z = 4 \times 2 \times 2$. (c) 3D positional embedding. Separate positional information along r, ϕ, z is concatenated into a single embedding. Here, D is the projection dimension.

of incidence ranging from 70 to 90 degrees and the energy of the particle as 64, 128 & 256 GeV. Hence, our dataset is around 90k samples for training and 10k for validation, with each shower containing 40,500 voxels. These voxels form a 3D image with dimension 18×50×45, respectively in r, ϕ, z direction as shown in Figure 1b.

Metrics. For evaluating the accuracy of the showers from the model, we compare various physics-based shower observables such as lateral and longitudinal profiles of the shower, moments of those profiles, distribution of voxel energies, the total energy of the showers & per layer, etc. of generated showers with respect to Geant4's showers [14]. Note that in this paper, 'generation' refers to the creation of new showers, and 'reconstruction' refers to getting back the same output as the input in the model training process.

Preprocessing. To prepare the input data for the model, we divide the voxel energies (in MeV units) by the energy of the incident particle (in GeV units), which brings the input values between 0 & 1. This enables us to use sigmoid as the output activation function and binary cross entropy as the loss function.

4 Exploring the architecture

Transformers are permutation-invariant sequence-to-sequence models. This means the input needs to be in the form of a sequence and the position of each element of the sequence needs to be specified to the model. Hence, in order to perform a quick check, we conduct ablation studies around sequence formation and position information by means of Masked Language Modeling (MLM) [15]. Along with comparing shower observables, in order to validate the representations learned by the model, we do the task of predicting the energy and the angle of incident particles from those representations. Since the representations learned by the foundation models are information-rich, these representations can also be used for various downstream tasks in the future.

4.1 Predicting masked particle showers

In this section, we show the working of our model, TransformerMLM, to predict the hidden information. That is, given a partially masked shower as input, the goal of the model is to reconstruct the original shower. The framework is described as follows: the original shower, x is given as input and is split into N non-overlapping patches, x^0 to x^{N-1}, which is the required sequence. 75% of these patches are chosen randomly to be masked, i.e., the energy values are replaced by zeros. After masking (f_m), the patches are fed to the model (f_{mlm}). We use 4 encoder-only transformer blocks, 8 attention heads, 128 as projection dimension, and 256 nodes in the feed-forward block. There are projections of patches before (f_{p_1}) and after (f_{p_2}) the transformer blocks to match the projection dimension and patch dimension respectively. We also feed positional embeddings (pos^i) along with the projected patch to the model. More details on creating patches and positional embeddings are described in the following section. Denoting the model as f, the output of the model is:

$$\hat{x}^i = \sigma(f_{p_2}(f_{mlm}(f_{p_1}(f_m(x^i)) + pos^i))$$

In learning to predict the original showers from partial showers, the model learns the representation of the input data. Note that this is not a generative model, as it involves reconstruction based on partial shower.

4.2 Experiments and Results

Patches. Since transformers need a sequence as an input, we construct non-overlapping 3D patches from a 3D shower image, experimenting with different numbers of patches along each of the r, ϕ, z directions (Figure 1b). The observation is that the more the number of patches in any direction, the better. However, this comes at the cost of increased computational requirements. Hence, we conclude with $\Delta r \times \Delta\phi \times \Delta z = 18 \times 10 \times 3$ as the patch size and use this as default for all experiments.

Positional encoding. Unlike recurrent neural networks, transformers can process multiple elements of the input sequence in parallel. Since the entire sequence is processed in parallel, we need to inject the position of each element in addition to the element itself. We exploit this nature to feed 3D positions to the model. In fact, we concatenate three separate positional embeddings for each direction as shown in Figure 1c, where each positional embedding uses different frequencies of sine-cosine functions [20]. We also explore other types of positional embeddings such as learnable 1D & 3D, and fixed 1D positional embeddings. We find that the fixed 3D positional embedding works the best, and therefore, we use it as the default in experiments. Additionally, we experimented with having a rollover in ϕ, but it did not improve the results significantly.

Table 1: Accuracy (%) for predicting E & θ for different inputs. The first row establishes a baseline by directly feeding the shower to the classifier.

Input	Predicting E	Predicting θ
Shower	60	33
MLP-based VAE representation	99	65
TransformerMLM representation	**99**	**99**

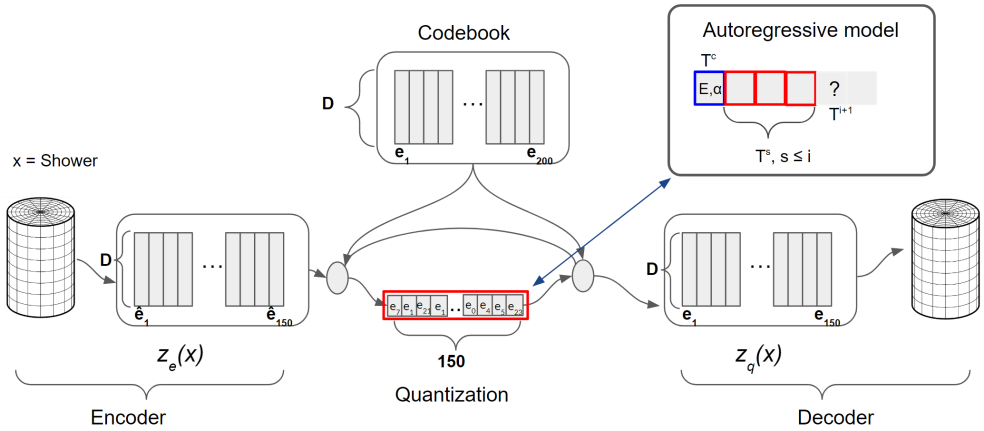


Figure 2: The figure shows the components of VQ-VAE and AR (top-right). The input shower is tokenized using the VQ-VAE's encoder and quantized by referring to the codebook. The tokenized representation of the shower is then fed to VQ-VAE's decoder to get the shower. AR models the tokenized shower's space (red tokens) separately in an autoregressive manner based on E & θ (blue token). During inference, VQ-VAE's encoder is not used.

How are the representations learned by the transformer? Once the model is trained, we use the learned representations to predict the initial conditions, E & θ to validate the training of the model. We use a 3-layer multi-layer perceptron as a classifier for predicting these initial conditions given representations from different sources, as shown in Table 1. The results conclude that the representations learned by the transformer model are indeed information-rich.

5 Generative Model

In this section, we describe our generative model capable of generating particle showers. It is a two-stage model inspired by Dall-E [22], where each stage is trained separately. The first stage is a Vector Quantized Variational Autoencoder (VQ-VAE) [23] used to tokenize the input showers, and the second stage is an Autoregressive (AR) model to generate these tokens. The reason for using a two-stage model is that VQ-VAE shortens the sequence length in the process of tokenization, thus reducing the computation cost for the attention mechanism in AR. AR models the high-level features and VQ-VAE models the low-level ones. The architecture is shown in Figure 2 and we explain both in the following subsections.

5.1 Vector Quantized Variational Autoencoder

Unlike VAEs, VQ-VAE has a discrete latent space represented by a finite set of learnable entities called codebook vectors (or tokens) with dimension D. The set of codebook vectors to represent a single input is chosen from a superset called codebook. The output from the encoder is quantized using a nearest-neighbour search on the codebook to obtain the discrete latent space. The loss function for the VQ-VAE contains additional terms to bring the codebook vectors and output of the encoder close together, and is as follows:

$$L_{vq} = L_{reconstruction} + ||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2,$$

Here $z_e(x)$ is the output of the encoder, e is the token embedding, β is the commitment loss weight, and sg is the stop-gradient operator. The process of patching the shower, projection of patches, and adding positional embeddings is the same as described in section 4. This is then passed to the transformer blocks, where we use 2 encoder-only blocks in the encoder and 2 encoder-only blocks in the decoder with 4 attention heads in each block. Before the latent layer, the patches in $z_e(x)$ are concatenated and then linearly projected to match the dimension of the entire latent space. The opposite is applied to connect the quantized latent space, $z_q(x)$ to the decoder.

5.2 Autoregressive model

Once the VQ-VAE is trained, we can encode and decode the shower to and from the latent space. However, due to a discrete latent space in VQ-VAE and its unknown probability distribution, it is not possible to directly sample the tokens, hence generating new showers. Therefore, we use an AR model to learn the latent space of VQ-VAE. The goal of the AR model is to autoregressively predict the tokens in the latent space. It is trained as a classification model (f_{AR}), which outputs the probabilities over the next token (T^{i+1}) given the previous tokens and mask, m as follows:

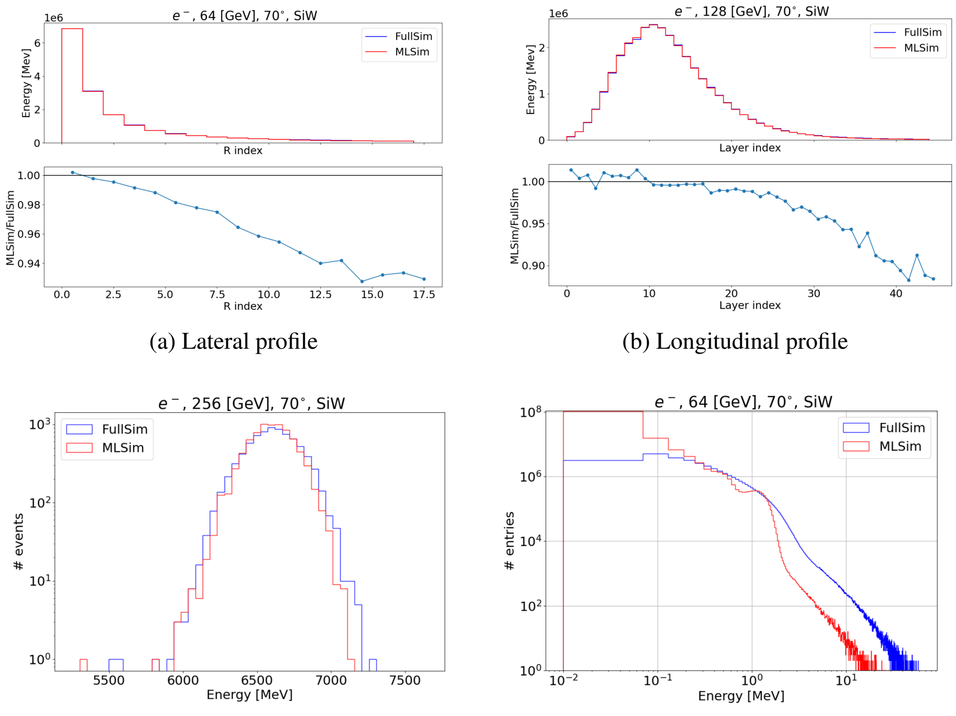
$$p(T^{i+1}) = f_{AR}(T^c, T^0, ..., T^i, m)$$

Since the goal is to have a conditional generation, the first token is a condition token (T^c) obtained by passing the conditions (E & θ) through a linear projection. The mask is fed to the model to prevent it from conditioning on future tokens. We sample a new token from the obtained probabilities and the process continues till we get all the necessary tokens, 150 in our case. This process of stochastic sampling makes the AR a generative model. AR consists of 4 decoder-only transformer blocks with 8 attention heads each.

Generation of new showers. First, the condition token is created, followed by utilizing the AR model to predict all the remaining tokens. These tokens are then passed to the VQ-VAE decoder to generate the shower. For the adaptation of the generative model to a new detector, we hypothesize that the VQ-VAE will attain the robustness required to tokenize any type of shower with more data, necessitating only fine-tuning the AR to the desired detector. However, validation of this hypothesis remains crucial.

5.3 Results

As mentioned in Section 3, we compare various shower observables for showers obtained from Geant4 (FullSim) and from our model (MLSim) as per the process described in Section 5.2. A selection of those shower observables for different energies of the incident particle is shown in Figure 3. We are able to get fair accuracy for most of the observables, albeit a bit of a mismatch at the tails, 7% for the lateral profile, and 10% for the longitudinal profile. This mismatch is not observed when doing the reconstruction by utilizing only the VQ-VAE, which means that this could be alleviated by incorporating techniques to overcome the class imbalance in AR. However, the accuracy of the voxel energy distribution is poor for generation (AR + VQ-VAE's decoder), and visible already at the reconstruction level (VQ-VAE only). We believe that this is due to the non-optimal combination of preprocessing and the output activation function, which presently results in skewed data distribution and less feedback from the sigmoid function as the input to the sigmoid approaches the tails. Improving the voxel energy distribution is one of the key tasks we are focusing on.



(a) Lateral profile

(b) Longitudinal profile

(c) Total energy per shower

(d) Voxel energy distribution

Figure 3: Shower observables for generated showers. The closer the GEANT4 (blue) plots to that of generated showers (red), the better.

6 Conclusion

In this paper, we propose a transformer-based generative model for the task of generating calorimeter showers, with the end goal of designing a detector-agnostic model that can be fine-tuned to any detector. At present, we introduce the first prototype of our model trained on a subset of data to validate the working of the model. We obtained promising preliminary results. We aim to improve the voxel energy distribution and to bridge the gap between the results of the VQ-VAE and the AR model. Besides these, one of the main future works is to conduct large-scale training, both in terms of model size as well as dataset size and diversity to analyze the generalization capability of the model.

Acknowledgements

This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments [24] and has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement No. 101004761.

References

[1] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand et al., Nuclear Instruments and Methods in Physics

Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506**, 250 (2003)

[2] J. Apostolakis, M. Asai, S. Banerjee, R. Bianchi, P. Canal, R. Cenci, J. Chapman, G. Corti, G. Cosmo, S. Easo et al. (HEP Software Foundation Collaboration) (2018), `1803.04165`

[3] D. Jang, pp. 2074–2080 (2009)

[4] Geant4 Collaboration (Geant4), *Par04 example*, `https://gitlab.cern.ch/geant4/geant4/-/tree/master/examples/extended/parameterisations/Par04`

[5] G. Aad, B. Abbott, D.C. Abbott, A.A. Abud, K. Abeling, D.K. Abhayasinghe, S.H. Abidi, A. Aboulhorma, H. Abramowicz, H. Abreu et al., Computing and Software for Big Science **6**, 7 (2022)

[6] O. Amram, K. Pedro (2023), `2308.03876`

[7] B. Käch, I. Melzer-Pellmann, arXiv preprint arXiv:2305.15254 (2023)

[8] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger, P. McKeown, arXiv preprint arXiv:2305.04847 (2023)

[9] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, K. Krüger, Computing and Software for Big Science **5**, 13 (2021)

[10] V. Mikuni, B. Nachman, Physical Review D **106**, 092009 (2022)

[11] C. Krause, D. Shih, arXiv preprint arXiv:2110.11377 (2021)

[12] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh, D. Shih, arXiv preprint arXiv:2302.11594 (2023)

[13] R. Bommasani, D.A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M.S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill et al., arXiv preprint arXiv:2108.07258 (2021)

[14] D. Salamani, A. Zaborowska, W. Pokorski, Physics Letters B **844**, 138079 (2023)

[15] J. Devlin, M.W. Chang, K. Lee, K.N. Toutanova (2018)

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., **33**, 1877 (2020)

[17] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, M. Chen, arXiv preprint arXiv:2204.06125 **1**, 3 (2022)

[18] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E.L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans et al., Advances in Neural Information Processing Systems **35**, 36479 (2022)

[19] S. Reed, K. Zolna, E. Parisotto, S.G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J.T. Springenberg et al., arXiv preprint arXiv:2205.06175 (2022)

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Advances in neural information processing systems **30** (2017)

[21] D. Salamani, A. Zaborowska, *High Granularity Electromagnetic Calorimeter Shower Images* (2022), `https://doi.org/10.5281/zenodo.6082201`

[22] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, I. Sutskever, **139**, 8821 (2021)

[23] A. Van Den Oord, O. Vinyals et al., Advances in neural information processing systems **30** (2017)

[24] M. Aleksa, C. Joram, P. Farthouat, A. Onnela, J. Blomer, C. Gargiulo, P. Janot, Strategic R&D; Programme on Technologies for Future Experiments (No. CERN-OPEN-2018-006). (2018)