# Transformers for Generalized Fast Shower Simulation

Renato Cardoso[1], Nadya Chernyavskaya[1], Kristina Jaruskova[1], Witek Pokorski[1], Piyush Raikwar[1], Dalila Salamani[1], Mudhakar Srivatsa[2], Kalliopi Tsolaki[1], Sofia Vallecorsa[1], Anna Zaborowska[1]

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

# Foundation models

*Realistic photo of wall-e on the streets of London*

[https://dalle2.gallery](https://dalle2.gallery)

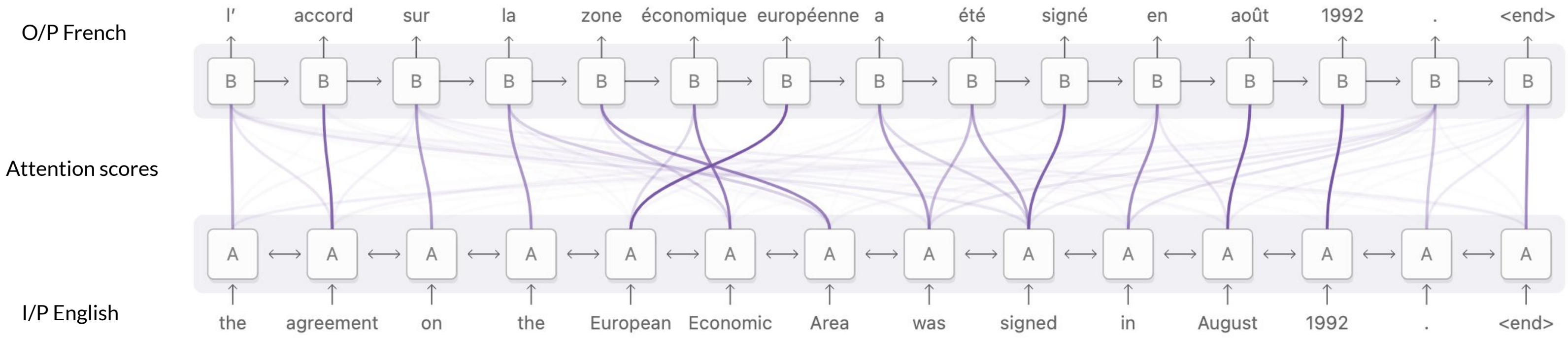
- The idea of foundation models started from very large pre-trained language models.
- Examples:
    - BERT, GPT-3, ChatGPT (Generative language models)
    - DALL-E, DALL-E 2, Imagen (Text to Image models)
- These models are typically **trained on very large & diverse datasets and variety of tasks** allowing them to learn patterns and represent common concepts and relationships.
- Generally, their architecture is **transformer-based**.

# Motivation

- Development of machine learning models for fast shower simulation is computationally expensive.
- Moreover, designing model for each experiment requires dedicated expertise.
- Therefore, **train once, then adapt** to new detectors, quickly.

- **Transformers** as building blocks in foundation models:
  - A **generalized architecture** that works with any type of data, e.g., text, images, audio, etc.
  - Models long-range dependencies (**Attention** mechanism).

# Attention in transformers

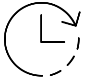



- **Dynamically focuses on important parts** in the input.
- Helps in modelling correlations between energy deposits.

# Our roadmap

✓ 1. Check if the transformers can learn good representations of our shower data.


2. Build a generative model for fast shower simulation.
   1. **Autoregressive**
   2. Diffusion

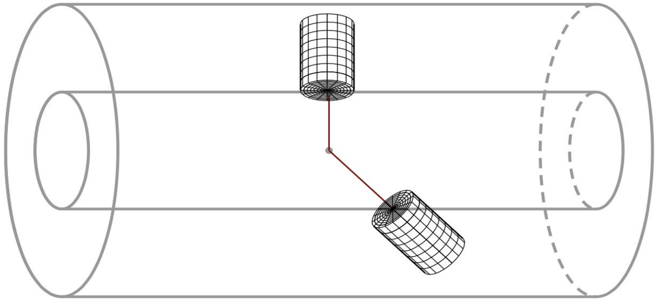


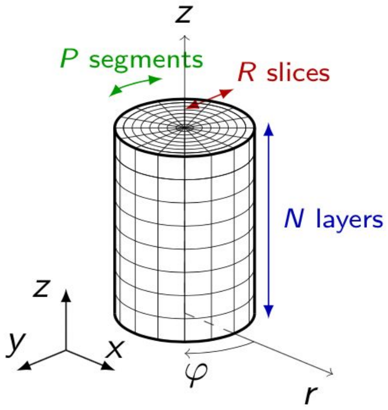
3. Scale in both model (size) and dataset (size & variety).

**Final goal - A generalizable foundation model for fast simulation adaptable to new data**

# Dataset

We utilize a dataset similar[1] to "CaloChallenge Dataset 3". (Talk at CHEP'23)





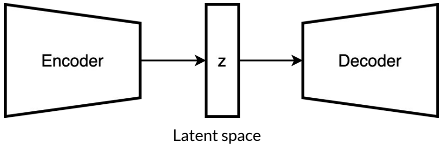

$$r \times \varphi \times z = 18 \times 50 \times 45$$

For the shown preliminary results, we use the following subset (~100k samples):

- Angle of incident $e^-$ = 70°, 80°, 90°
- Energy of incident $e^-$ = 64, 128, 256 GeV
- Sampling calorimeter with silicon and tungsten layers[2] (SiW)

[1] More incident angles and discrete energy spectrum
[2] Layer thickness: 0.3 mm + 1.4 mm for Si & W respectively

*Variational autoencoder (VAE)*




# Autoregressive model architecture

Two-stage model (both models have transformer-based architecture):

*Train first*

1. **Vector Quantized Variational Autoencoder (VQ-VAE)**
   - An autoencoder with discrete latent space.
   - Compresses and decompresses the shower to and from the latent space.
   - Thus, reduces the computational burden on the 2nd stage.

*Train later keeping VQVAE frozen*

2. **Autoregressive prior**
   - Unlike VAE, VQ-VAE cannot generate new samples[1].
   - Hence, an autoregressive prior to learn the latent space distribution.

[1] The latent space is discrete instead of Gaussian, thus not straightforward to sample from.

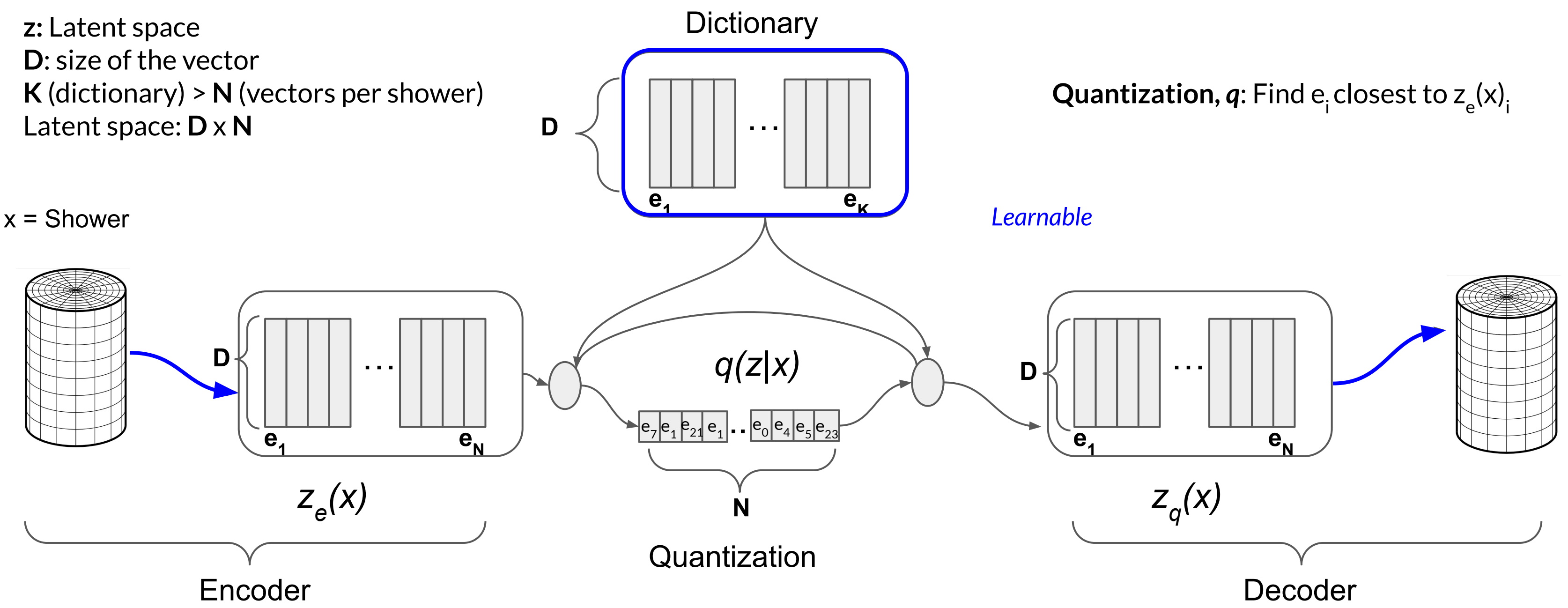
# VQ-VAE

Maps the input to and from a finite set of **vectors** (latent space).

- **z:** Latent space
- **D**: size of the vector
- **K** (dictionary) > **N** (vectors per shower)
- Latent space: **D** x **N**

**Quantization, *q***: Find $e_i$ closest to $z_e(x)_i$

*Learnable*



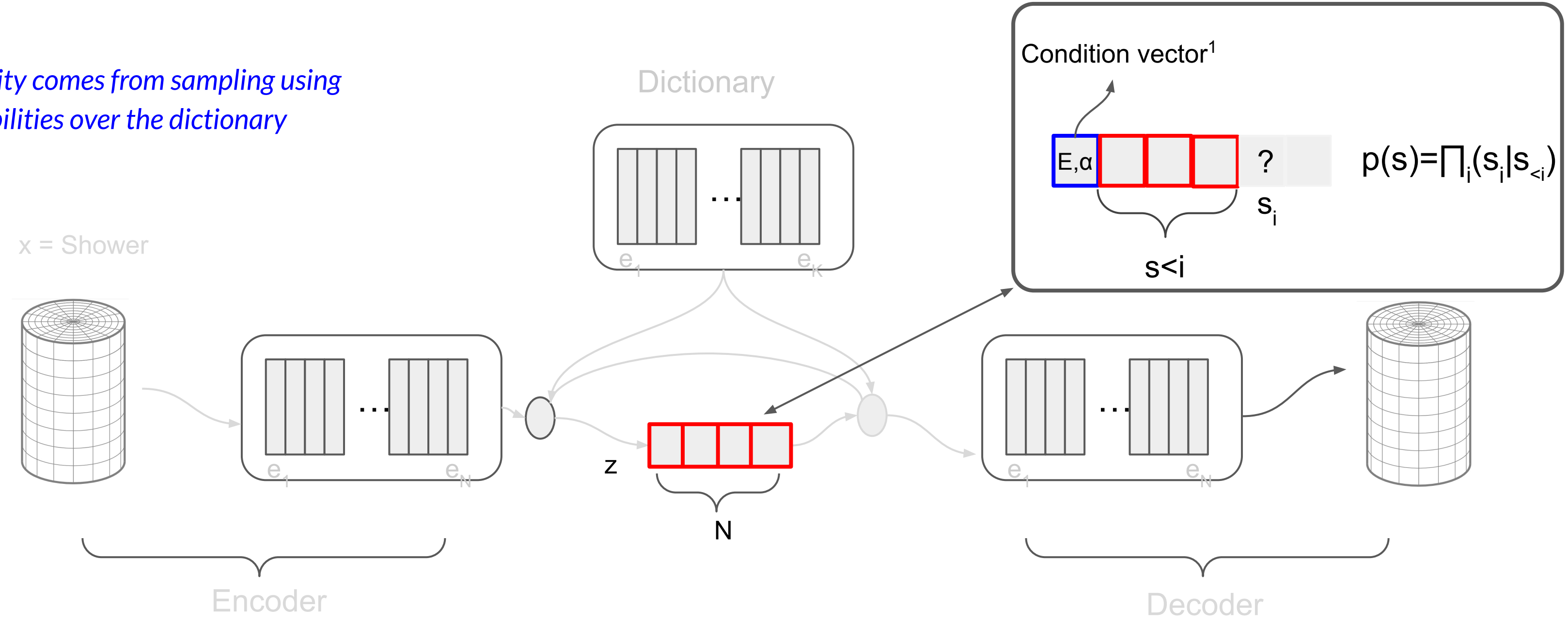

[Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018]

# Autoregressive prior

**Given previous vectors, predict the next vector.**

The goal is to mimic VQ-VAE's dictionary vector distribution.

*Stochasticity comes from sampling using the probabilities over the dictionary*



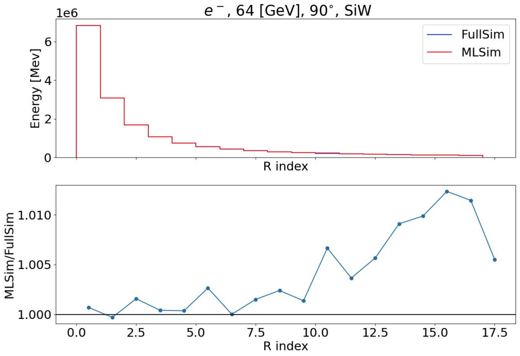

[1]Condition vector ([energy, angle, (+ detector, position offset)]) projected via a linear layer of dimension D.

# Generative model

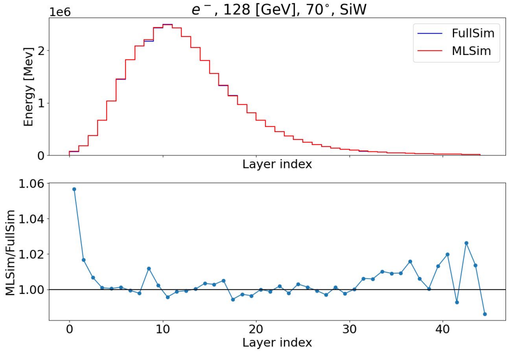

**Adaptation of generative model for new data:**

- Autoregressive prior is fine-tuned on the new detector’s data.
- We believe VQ-VAE (thus also dictionary) would become robust with more data and should remain frozen. (Needs to be investigated)

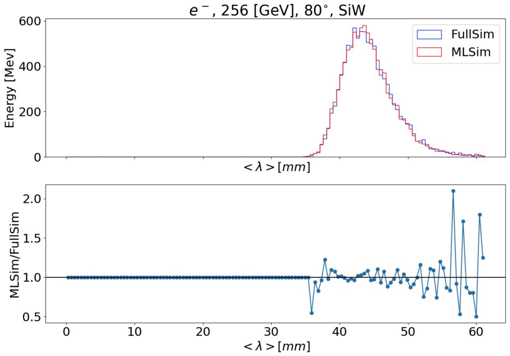
# Results - VQVAE




*Lateral profile*




*Longitudinal profile*

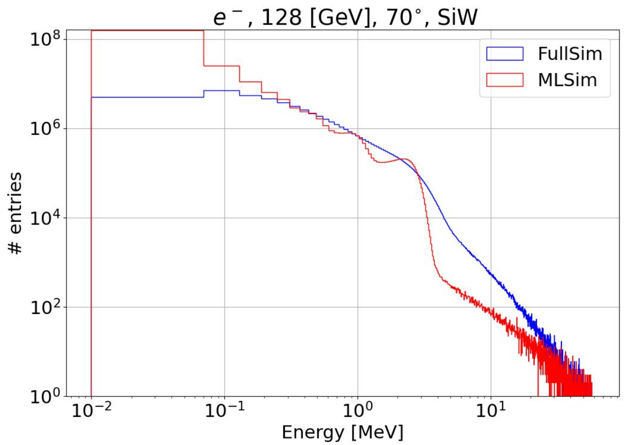



*Longitudinal first moment*

VQ-VAE was able to model lateral & longitudinal profiles,

first & second moments really well.

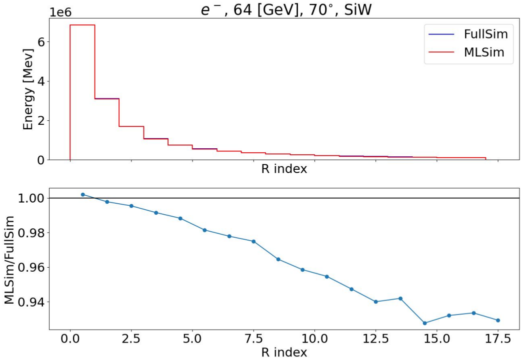
# Results - VQVAE



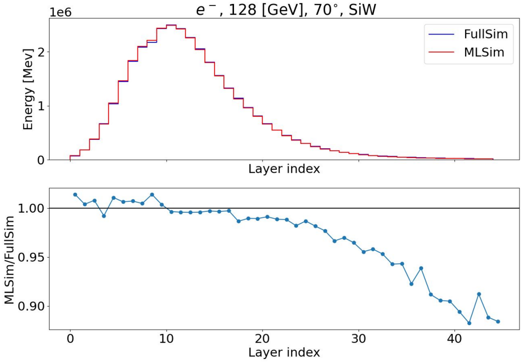

*Cell energy distribution*

- Accurate modelling of cell energy distribution is in progress. Currently leads to *blurry showers.*
- Introducing a GAN discriminator should help in properly modelling the cell energy distribution. (Next step)
- This also limits the autoregressive prior as VQ-VAE acts like an upper-bound.

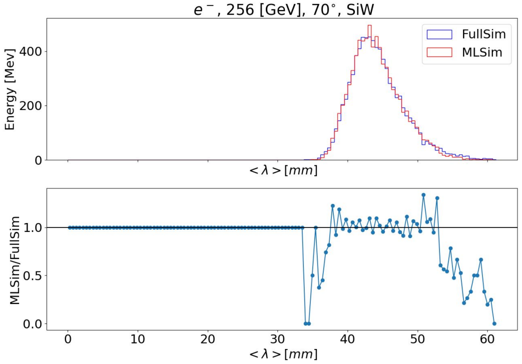
# Results - Autoregressive prior




*Lateral profile*




*Longitudinal profile*




*Longitudinal first moment*

- Autoregressive prior mimics the VQ-VAE vector distribution fairly well.
- The longitudinal & lateral profiles deviate at the tail due to uneven distribution of dictionary vectors.
- This should be overcome by using standard tricks to improve any classification model. (Next step)

# Conclusion

- Proposed a transformer-based generative model for fast simulation.
- This is a work in progress and we obtained promising preliminary results.
- We have several potential ideas to improve VQ-VAE and Autoregressive prior, e.g., GAN discriminator, Gumbel-Softmax quantizer, multi-scale architectures, which are under investigation.
- In parallel, we are exploring the diffusion model which has proven to be promising for images.
- One of the main future work is to conduct a large scale training and analyze the generalization capability of the model.

## Thank you for listening!

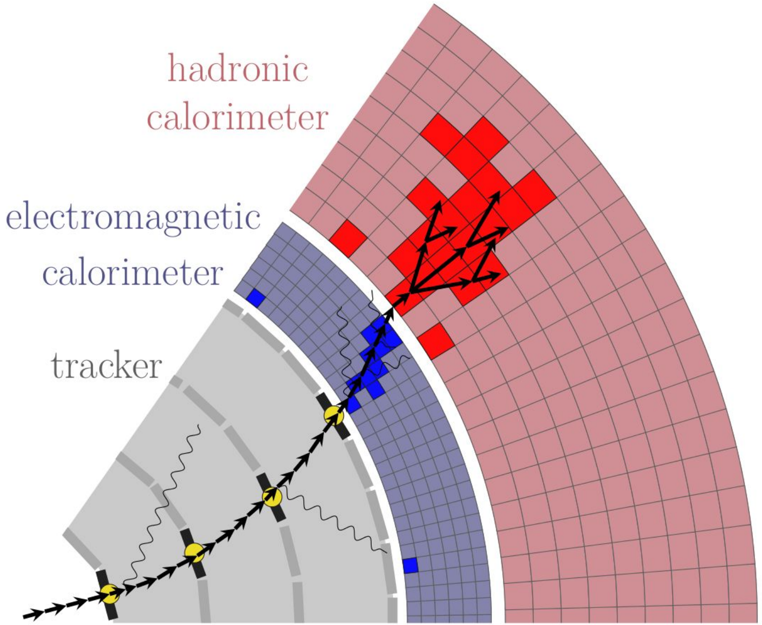
## Questions?

piyush.raikwar@cern.ch

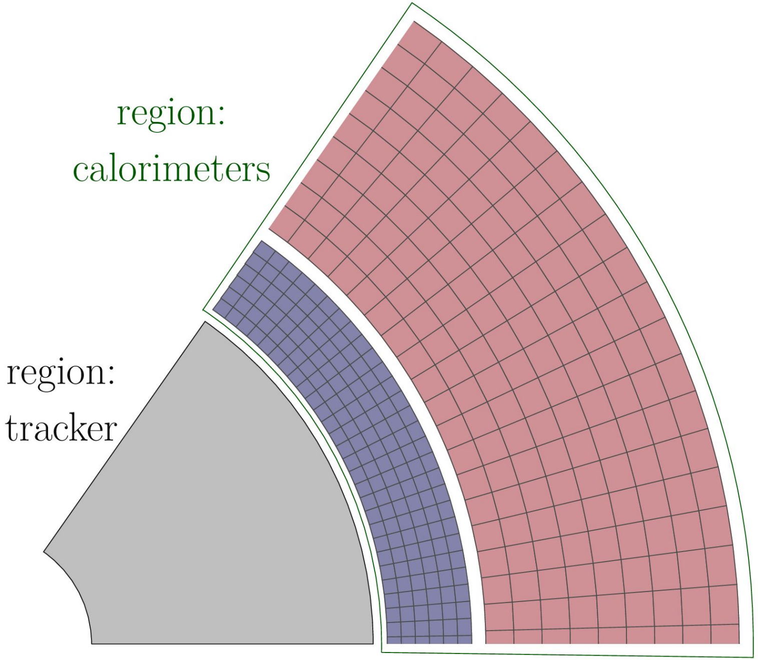
# Backup

# Fast shower simulation

FullSim




FastSim

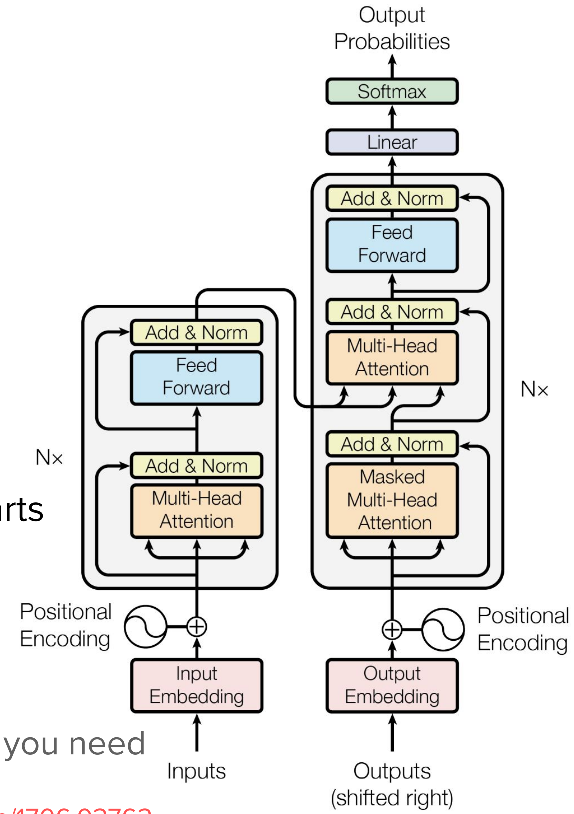


# Dataset

- High Granularity Electromagnetic Calorimeter Shower Images [zenodo]
  - Energy = 1 GeV - 1 TeV
  - Angle = 50° - 90°
  - Geometries = SiW, SciPb
  - ~10, 000 events each

# Transformer

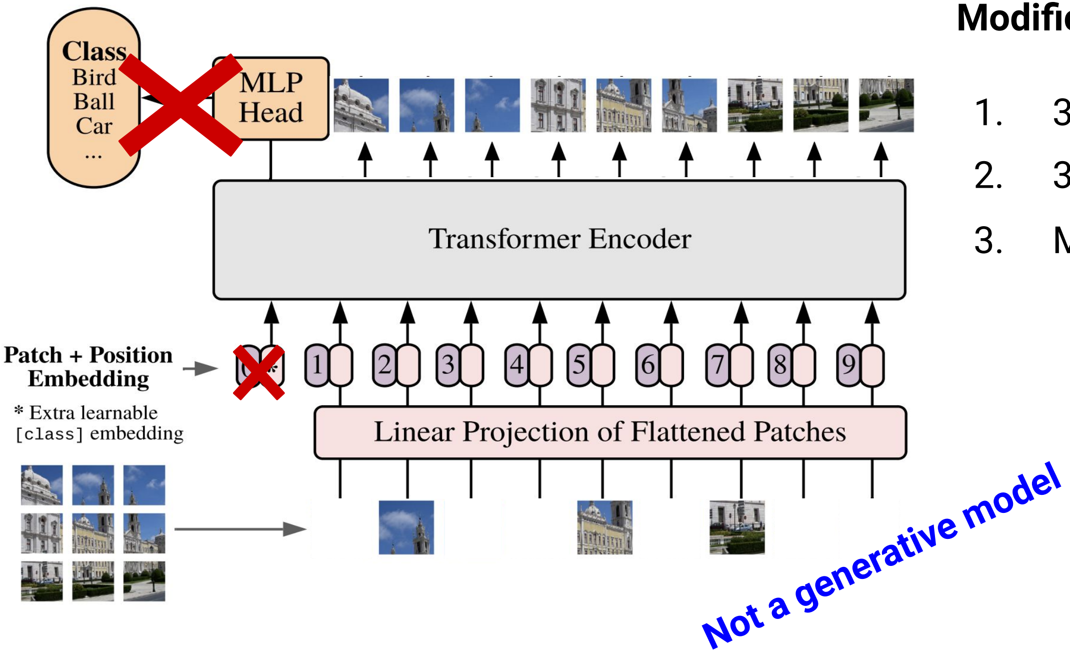
- Proposed for sequence-to-sequence tasks.
- I/O is any type of sequences.
- Encoder-Decoder blocks.
- Positional embeddings.
- **Attention:** Dynamically focus on important parts in the input.
- Multi-headed attention.




Attention is all you need

https://arxiv.org/abs/1706.03762

# Self-supervised training




**Modifications to ViT (Vision transformers)**

1. 3D image, 3d patches
2. 3D positional embeddings
3. Masked language modelling (MLM)
    a. Remove “MLP Head”
    b. Remove “class embedding”
    c. Add masking
    d. Reconstruct original image

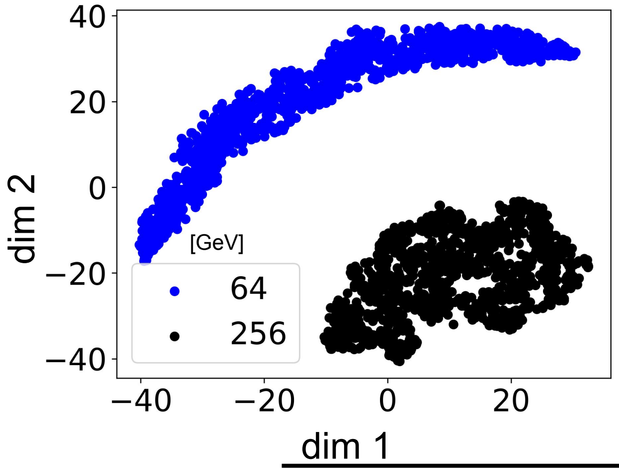
**Masked language modelling (MLM) is learning representations by trying to predict hidden information.**

# Checking representations

Q : How to validate that the transformer model is learning a good representation of our shower data?

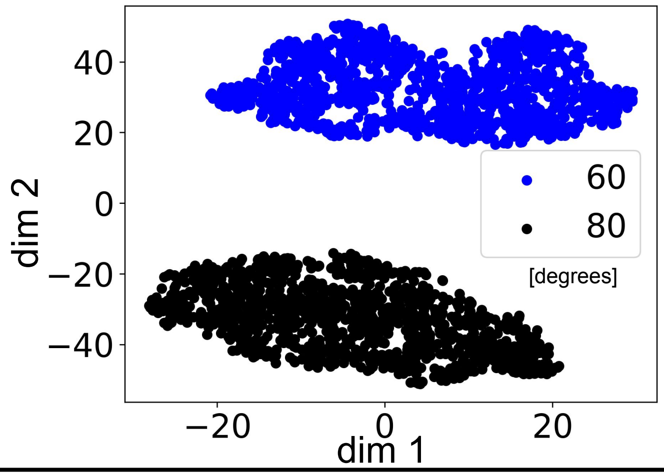
A : Use a "fake" downstream task: predict the energy/angle of the incoming particle using the transformer's representation

Energy prediction




**99% accuracy**

Angle prediction

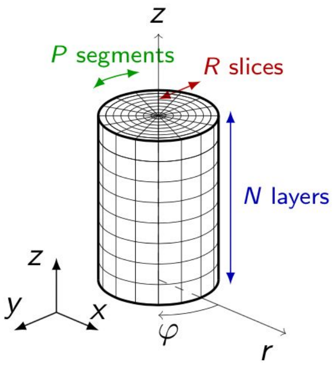



**99% accuracy**

# From shower to 3d sequence

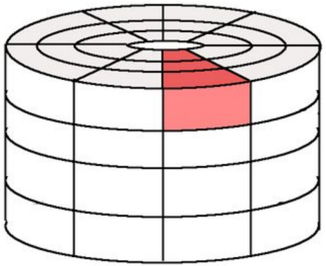
Transformers needs the input to be in the form of a sequence. Therefore,

- Patches are formed by making splits in r, $\varphi$ and z direction
- Patch configuration: 1 patch in r, 10 in $\varphi$ and 15 in z




1 shower = a set of patches

* Figure not drawn to scale



$r \times \varphi \times z = 18 \times 50 \times 45$

Patch size $r \times \varphi \times z = 18 \times 5 \times 3$

# Positional embeddings

Transformers are permutation invariant. Positional embeddings gives an understanding of position to the model.

**Explored**

- 1D learnable keras embedding layer.
- Fixed 3D positional embeddings
  - Alternate sine-cosine.
  - Each direction takes $1/3^{rd}$ of the embedding dimensions.
- Phi-rollover

**Observation**

- Fixed 3D positional embeddings perform better *(default)*.

# Preprocessing & Loss function

**Preprocessing**

Division by energy value of the incident particle.

**Loss function**

- **VQVAE:** Binary crossentropy + VQVAE specific losses
- **Autoregressive prior:** Crossentropy

# Two stages

- VQ-VAE is not a generative model (discrete latent space).
- Hence, needs autoregressive prior to model to learn the latent space.
- Autoregressive prior due to sampling is a generative model.
- Autoregressive prior cannot be used alone:
  - It needs to predict a class. We have continuous energy deposits.
  - Sequence (voxels) would be too long.
- Since autoregressive prior needs to predict a class, it needs a discrete (finite) latent space from the autoencoder. Hence, VQ-VAE over VAE.
- TLDR - **both VQ-VAE and autoregressive prior depends on one another**.