Latent Generative Models for Calo Sim

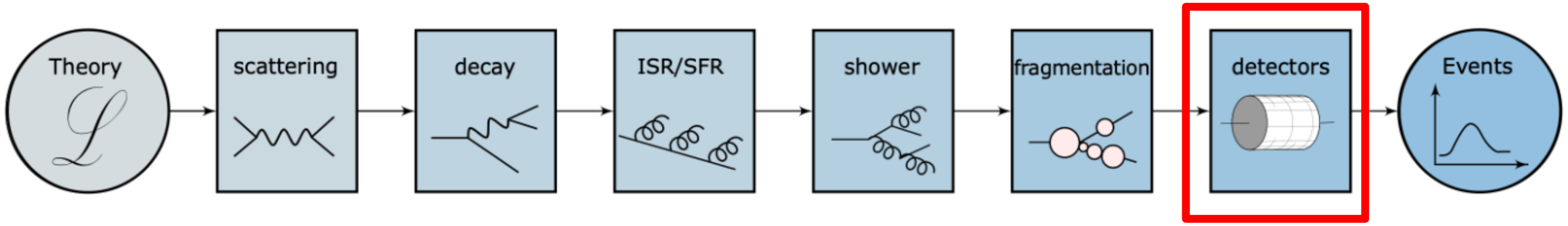
Chase Shimmin[1], Qibin Liu[2,3], Xiulong Liu[3],

Eli Shlizerman[3], Shih-Chieh Hsu[3]

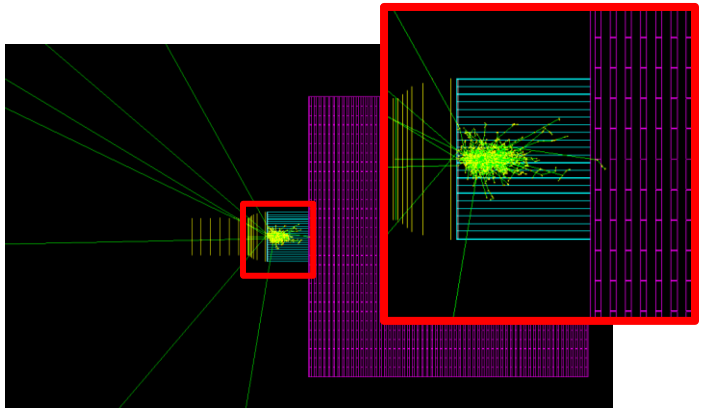
1 Yale University
2 TDLI., Shanghai Jiao Tong University
3 University of Washington

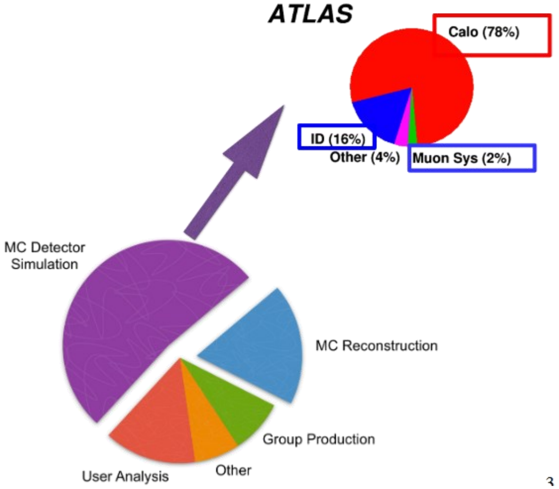
Introduction

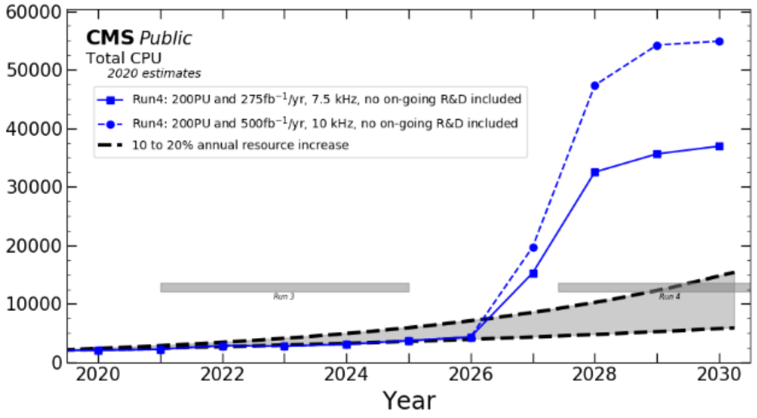
- Calorimeter: vital detector measuring the particle energy in high energy physics
- Simulation of calorimeter: important but challenging
  - Measure the basic physics information particle energy
  - Need tracing of huge amount of secondary particles (`shower`)
- More than 75% of simulation time spent on the calorimeter in ATLAS
- Fast simulation of calorimeter mostly needed and necessary for next generation experiment



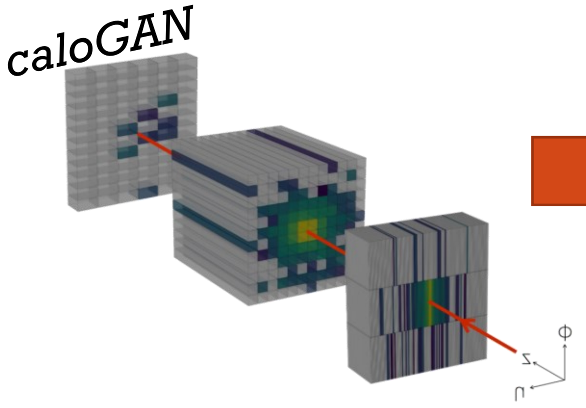
Machine Learning and LHC Event Generation, A. Butter et al. [2203.07460]



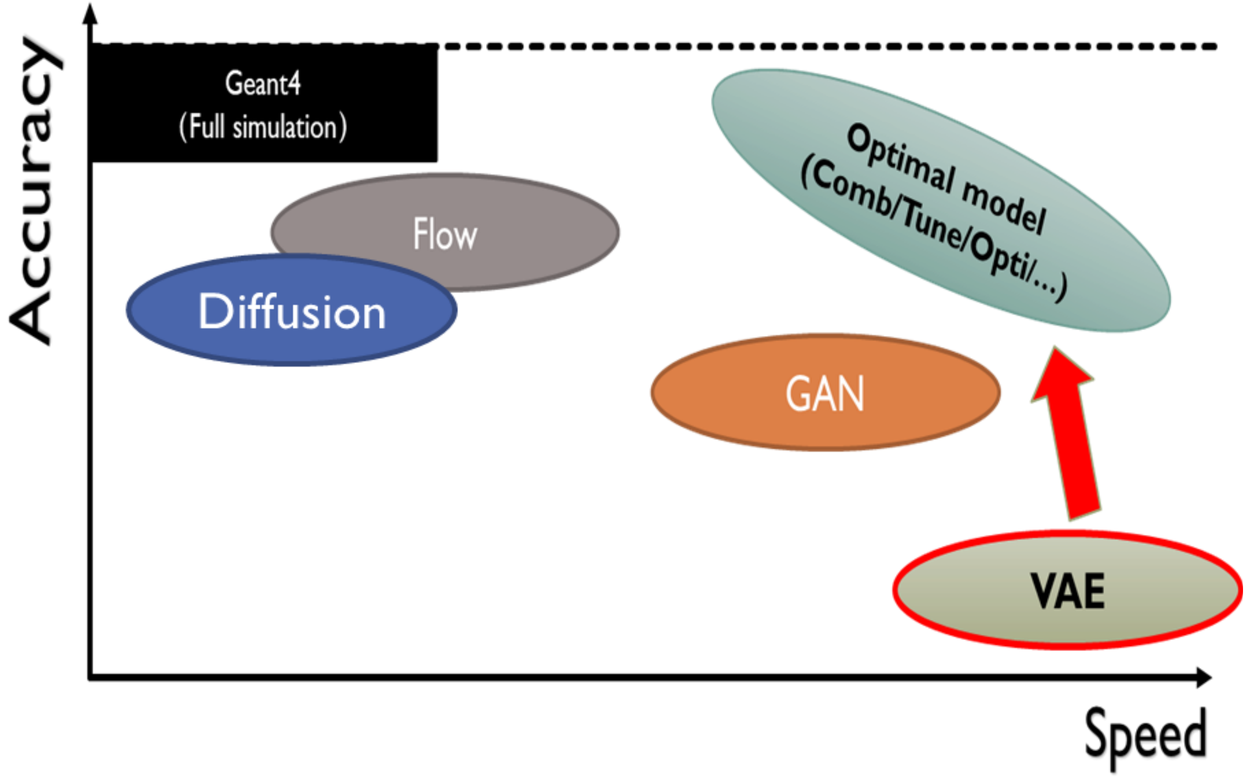




Machine Learning Methods for Fast CaloSim



$P(\overrightarrow{E_i}|E_{inc}, type, \dots)$



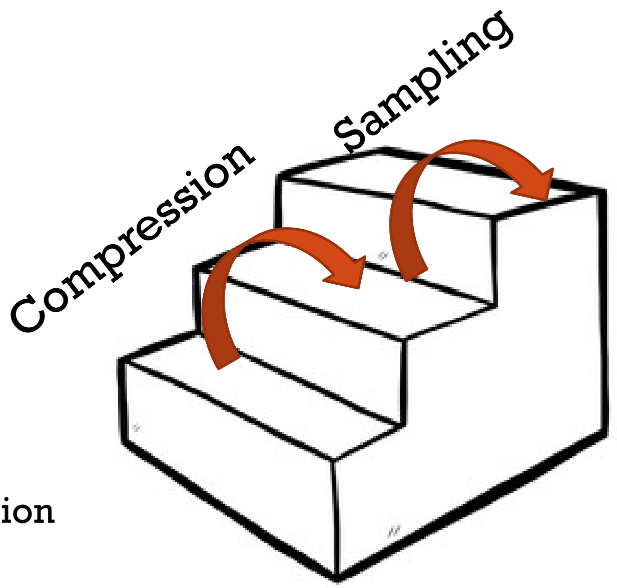
- Traditional method (e.g. parameterization)
  - Risk of over-simplification
  - Still rely on explicitly particle simulation
- Machine learning method
  - Treat calorimeter response as image or other formats
  - Directly learn the distribution and do the sampling
  - No tracing secondary particles
- Various DGM (deep generative model) available:
  - Flow: accurate but usually slow sampling speed
  - Diffusion: slow but promising at accuracy and scalability
  - GAN: fast but hard to train and tune
  - VAE: fastest but usually limited by the accuracy
- Why VAE-like methods:
  - Unique latent space enable latent generative model

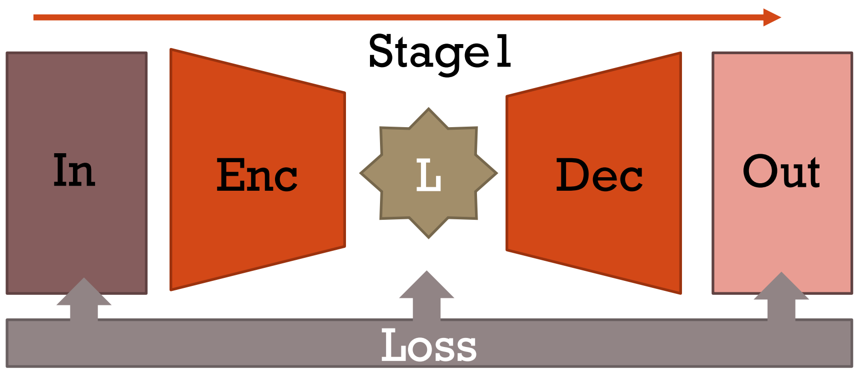
→ **Two-stages Generative Model**

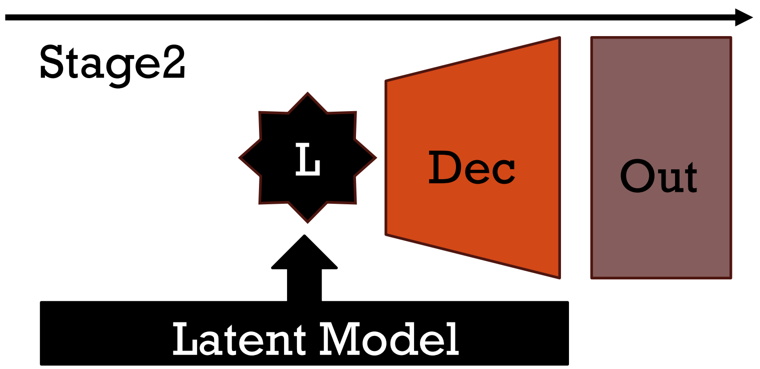
Disclaimer: not a serious comparison just for illustration of typical performance

Two-stages Generative Model

- Challenges of DGMs:
  - Making big model considering both training time and tuning effort
  - Generalization and migration between different dataset (format)
- Two stages calorimeter generative model:
  - Inspired by the VQVAE[1] and stable-diffusion[2]
  - Stage1: AutoEncoder-like model for compression
    - Flexibility: adapted to different geometry and aim to extract the common (physics) information
    - Well defined target: compress as much information into latent and recover it
    - Speed: fast training and forward pass
  - Stage2: Latent model for sampling and other tasks
    - Beneficial from mature model in other fields, e.g. cutting edge LLM
    - Explainable and physics embeddings, such as equivariant and symmetry
- We will present specific model on the calo challenge dataset

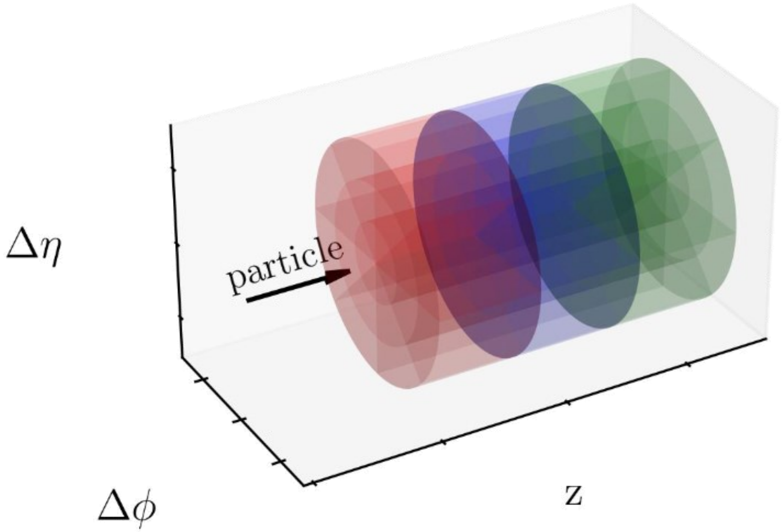


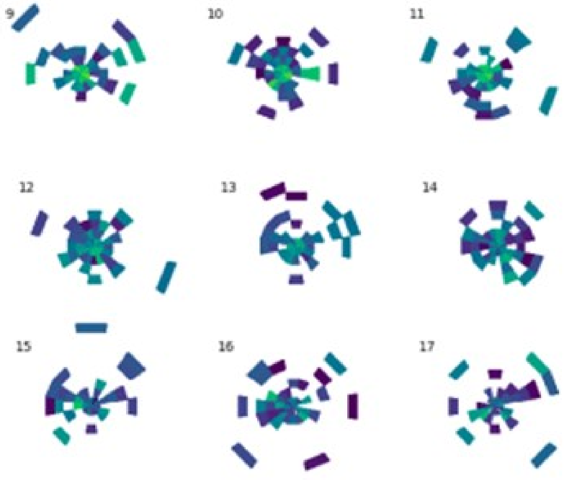




Recap of Calorimeter Dataset (and terminology)

- Cylindrical shape with **A**lpha, **R** and **Z** three dimension
  - Not uniformly correspond to the physical distance
  - Also possible to convert into $\Delta\eta, \Delta\phi$ and Z coordinate
    - Where the metric scores are evaluated
- Special "3D calo-image"
  - Represent the calorimeter response of the incident particle
  - Energy in each detector cell/pixel: E_i
  - Sum of all the energy in N-th Z layer: E_{layerN} and sum all E_{tot}
  - Incident energy of particle: E_{inc} (treated as condition)
- Dataset1: irregular geometry
  - Layer different alpha and R segmentation (some with ring-like geo)
- Dataset2/3: more regular but special coordinate/granularity
  - (A * R * Z) 3 dimensional data
  - Rotational symmetry and particle always at center and perpendicular





Input Data

N

Normalization

Common pre-processing for all calo challenge datasets

Large dynamic Range and Solution: Normalization

- Large dynamic range due to the needs of simulation and underlying EM interaction
  - Dynamic range of condition
  - Dynamic range of all the pixel energy in a same particle shower

Normalizations

- Encoder pass
  ① Normalized by condition and then "R" (E_tot/E_inc): [1e-6,1e6] → [0,1]
  ② Transformed by log1p (with proper scaling and shift)
- Decoder pass:
  ① **Softmax in last decoder layer**: ensure output normalized in the sense of exponential
  ② Recover the energy with R (sampled with latent model) then E_inc (condition)

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

E_inc=20 GeV

1	2	1
1	8	2
1	1	1

Normalization of E_inc

0.05	0.10	0.05
0.05	0.40	0.10
0.05	0.05	0.05

Normalization of R

R=0.90

0.06	0.11	0.06
0.06	0.44	0.11
0.06	0.06	0.06

Logarithm

2.22	2.52	2.22
2.22	3.13	2.52
2.22	2.22	2.22

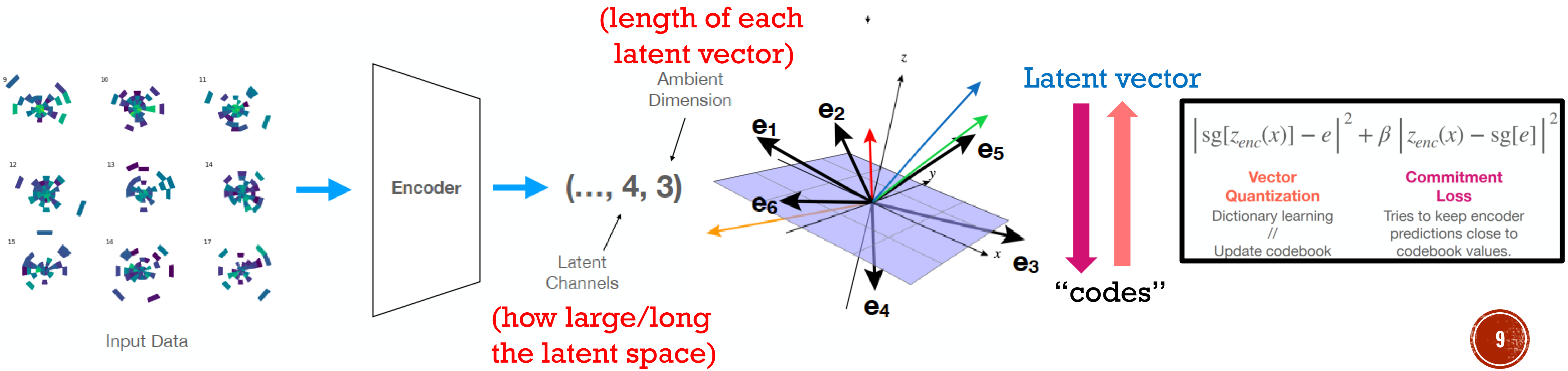
S1

STAGE1: VQVAE

Model to compress the specific dataset into latent space

Vector Quantized VAE

- Variant of VAE with additional latent regularization
- Quantize the latent (vectors) as discrete codes (finite number of "representative" vectors)
- Loss to drive the latent space well quantized besides of the typical AE loss (L1/L2)
  - **Quantization loss: codes faithfully represent the latent space**
  - **Commitment loss: encoded latent vectors close to codes**
- Extend the latent space to $\mathbb{Z}^L$ which is more flexible and descriptive



(length of each latent vector)

(how large/long the latent space)

Latent vector

"codes"

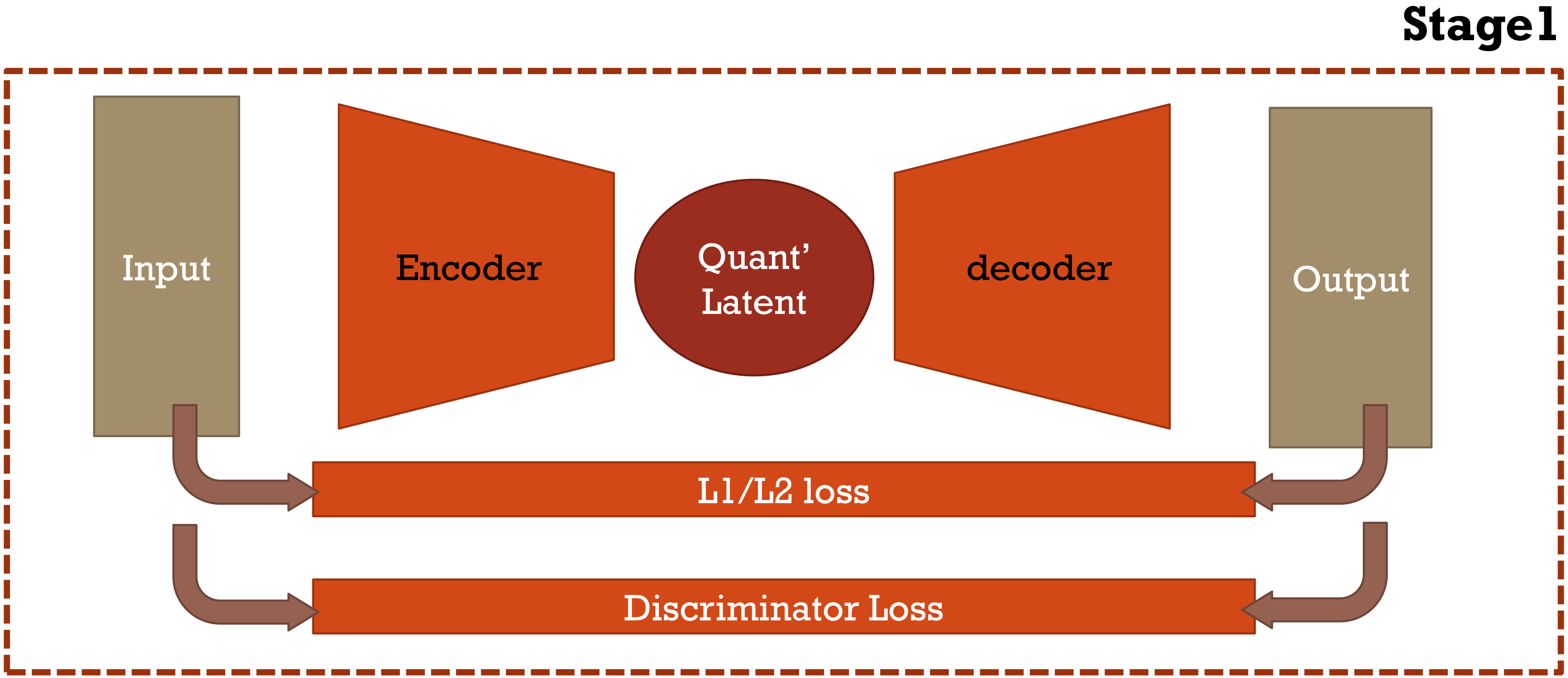
$$\left|\mathrm{sg}[z_{enc}(x)] - e\right|^2 + \beta\left|z_{enc}(x) - \mathrm{sg}[e]\right|^2$$

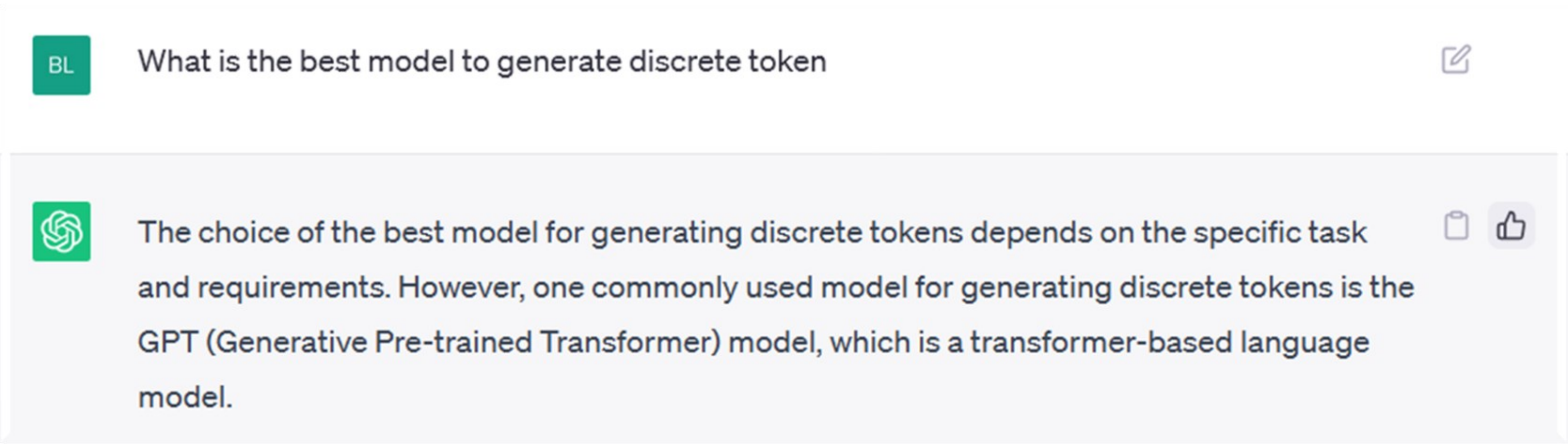
Vector Quantization
Dictionary learning
//
Update codebook

Commitment Loss
Tries to keep encoder predictions close to codebook values.

Design of Loss Functions

- Pixel wise Loss: L2 distance (MSE) between truth and decoded pixels
- Adversary Loss: discriminator loss and generator loss ~ similar to GAN
- Physics-perceptual Loss: (more on backups and not used in the submitted model)





What is the best model to generate discrete token

The choice of the best model for generating discrete tokens depends on the specific task and requirements. However, one commonly used model for generating discrete tokens is the GPT (Generative Pre-trained Transformer) model, which is a transformer-based language model.

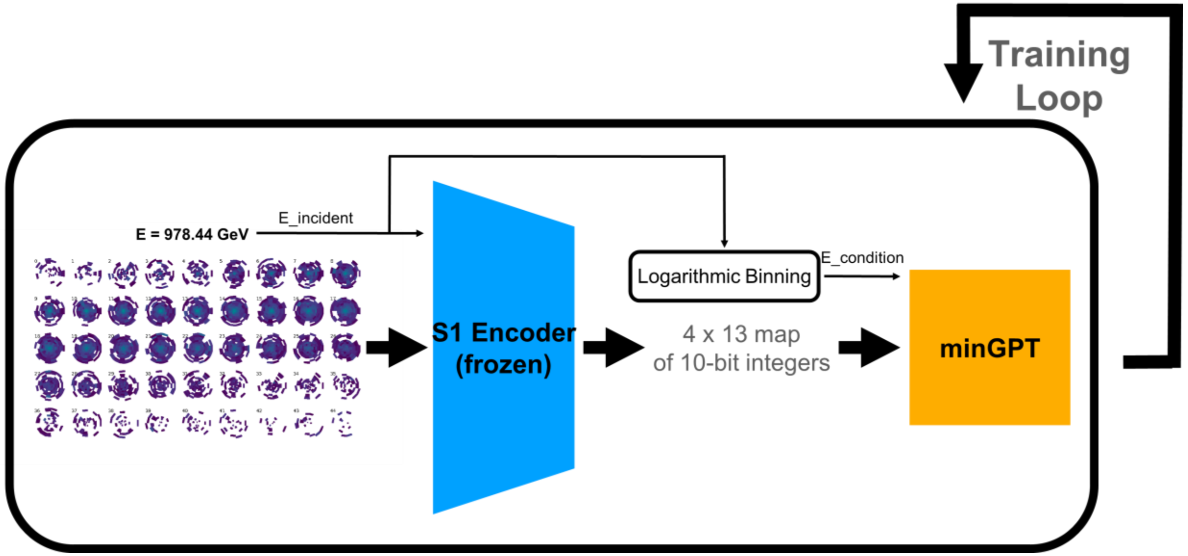
S2

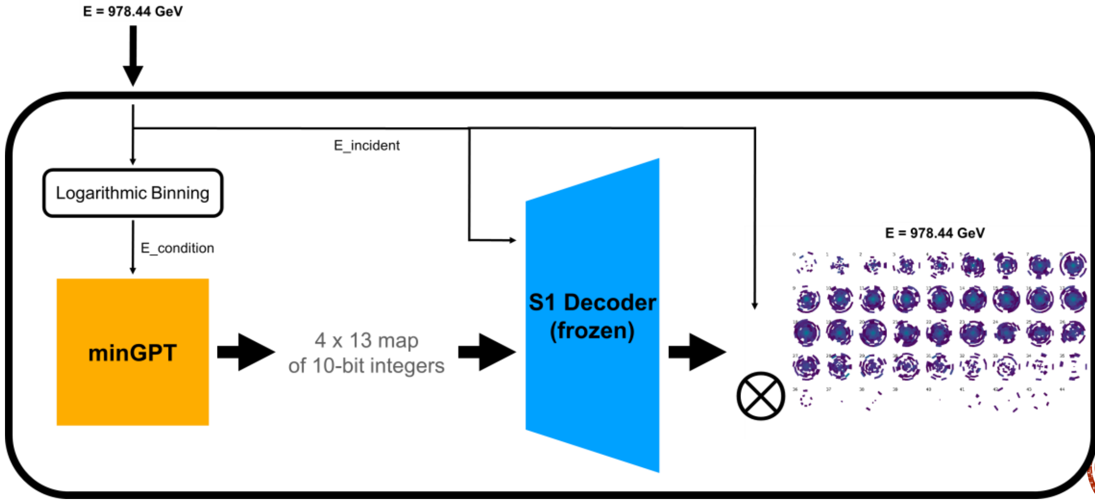
STAGE2: GPT

Conditional sampling of latent space

Latent Generation Model

- Latent space to be sampled: sequence of discrete numbers
- Similar to language model: codes ←→token
  - Latent discrete codes treated a language token
  - Sampling based on condition
- Various models(RNN, latent diffusion, GPT) tested and GPT is finally used
  - Implementation from minGPT [3]
  - General introduction of GPT model [ask GPT]



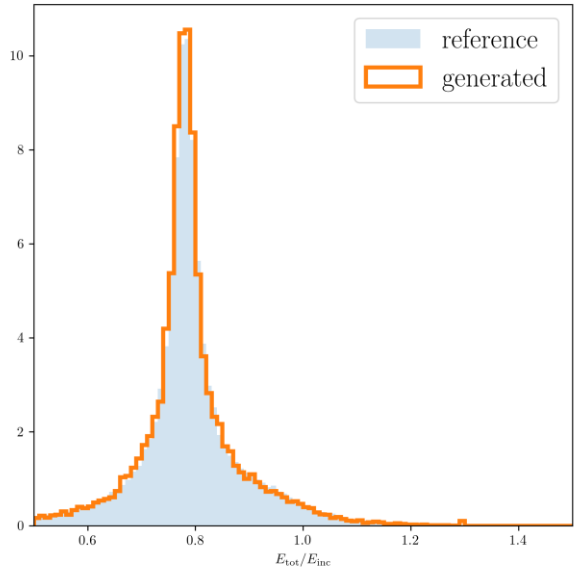


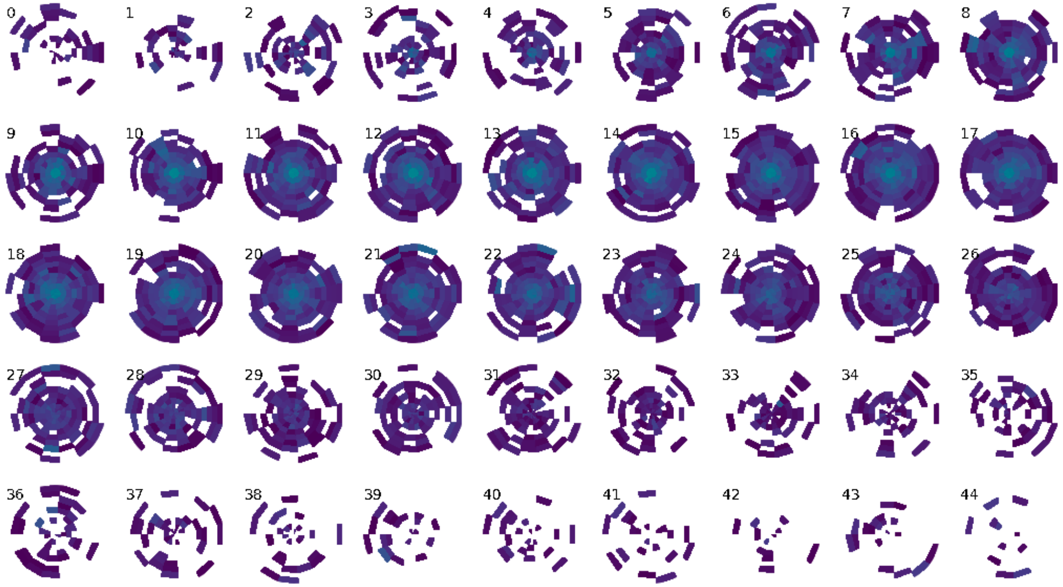
Generation of "R" (E_tot/E_inc)

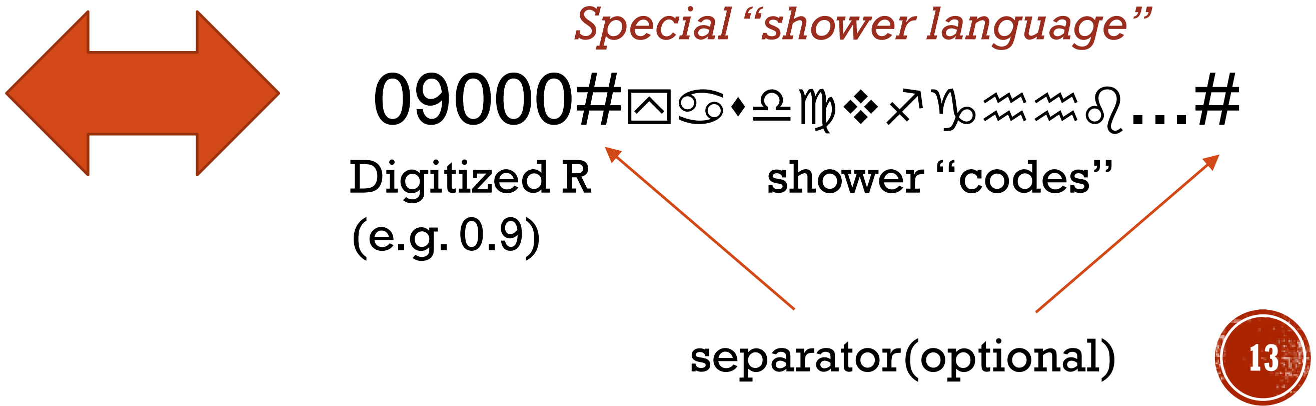
- Important feature of calorimeter dataset: E_tot != E_inc
  - Reflect the property of calorimeter design such as calibration and sampling ratio
  - Highly correlated to the condition, pixel energy, particle and detector design
  - Usually centralized (around 0.8) and with long tail

- Generate R in GPT:
  - Digitized to bits and treated as discrete inputs
  - Concatenate with latent sequence and sampling with GPT

$$R \equiv \frac{E_{tot}}{E_{inc}} = \frac{\sum E_i}{E_{inc}}$$







Special "shower language"

09000#⌧♋⬩♎♍❖♐♑♒♒♌...#

Digitized R (e.g. 0.9)

shower "codes"

separator(optional)

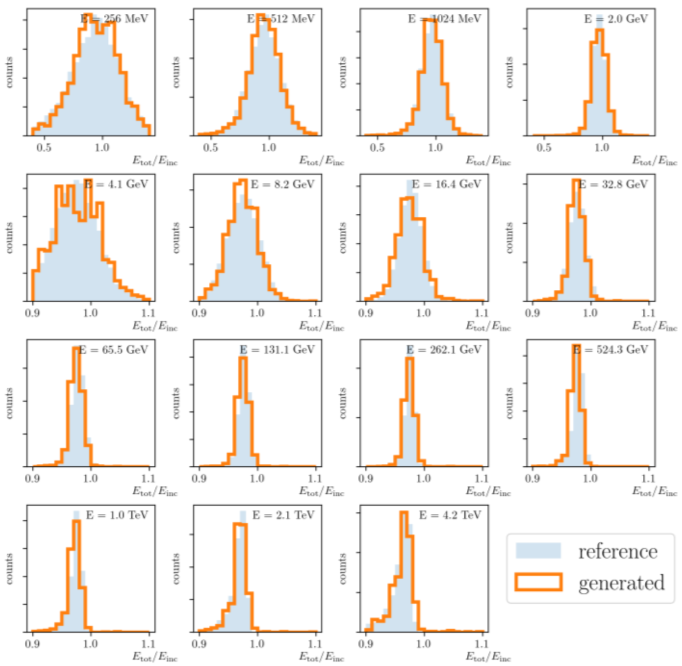
ds1

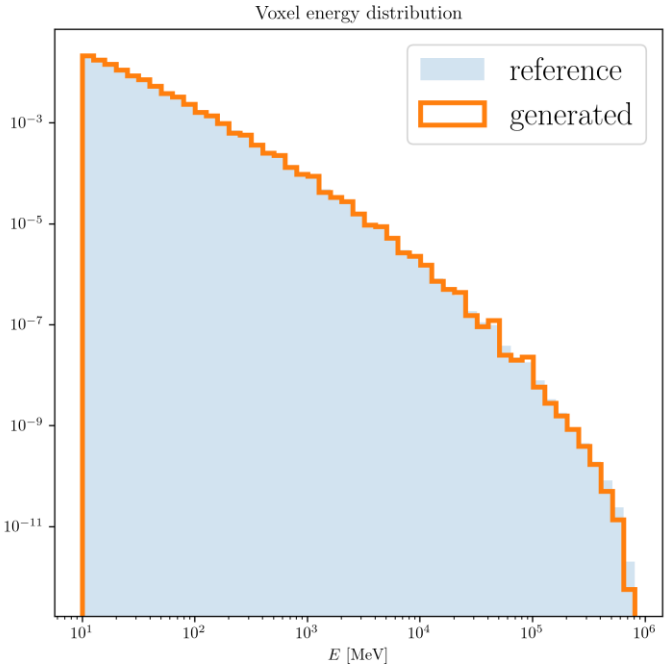
Dataset1

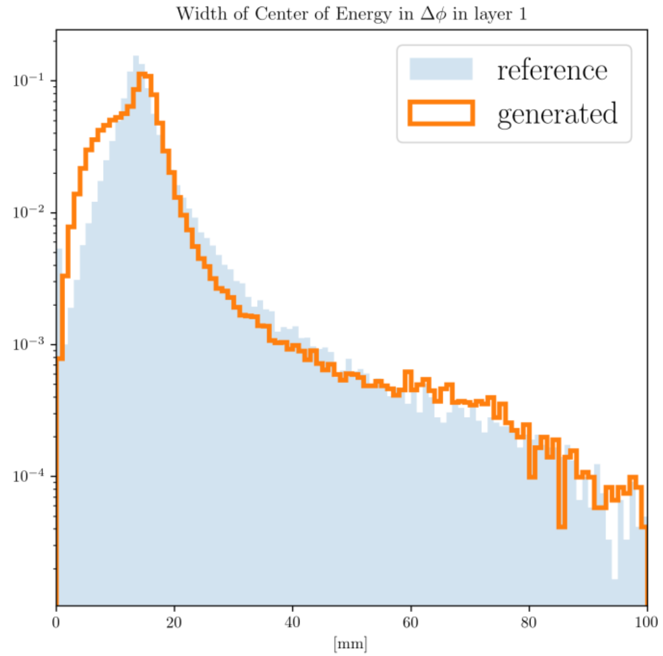
Generative Model Performance

DS1-photon

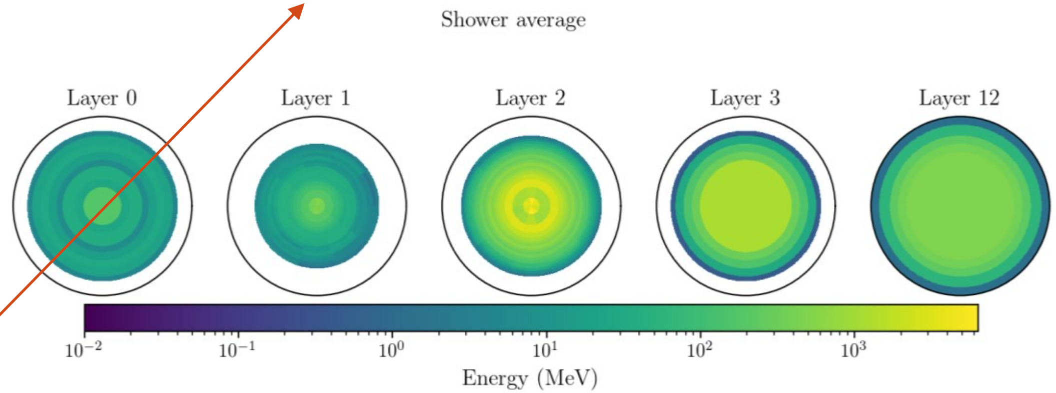
Thanks for the nice visualization from calo challenge!




Voxel energy distribution

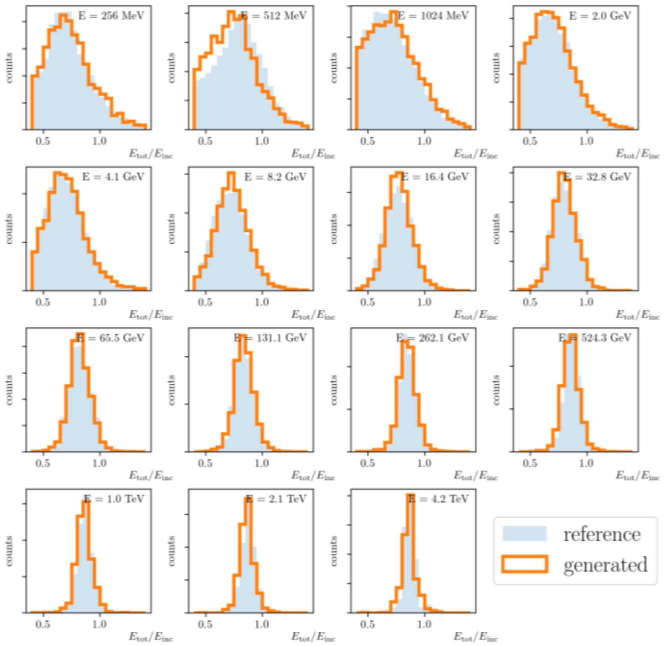

Width of Center of Energy in $\Delta\phi$ in layer 1

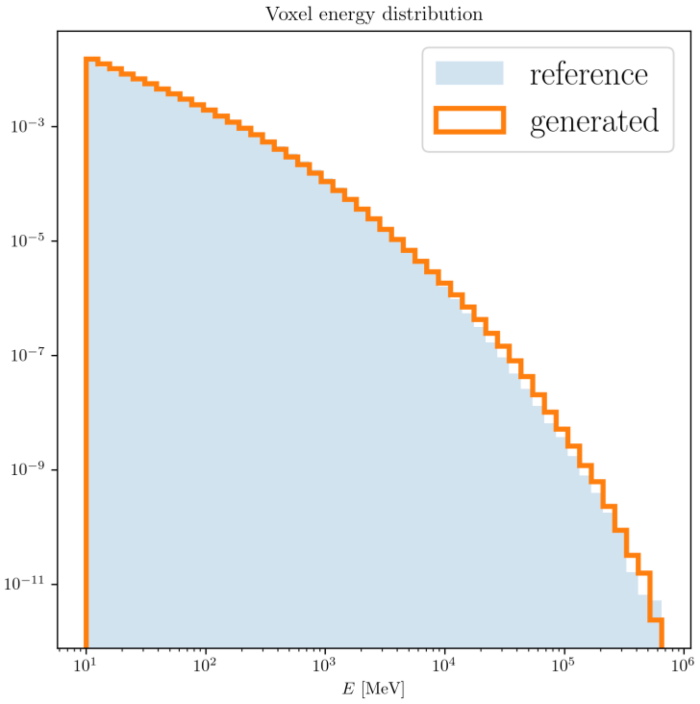
Scores Cat. (No.):	Mean	Best	Worst
Etot/Einc(16)	0.0146±0.0124	0.004	0.0542
Esum(5)	0.0054±0.0033	0.0028	0.0119
Eta_Center(2)	0.0154±0.0068	0.0085	0.0222
Phi_Center(2)	0.0053±0.0004	0.0049	0.0056
Eta_Width(2)	0.0400±0.0003	0.0397	0.0403
Phi_Width(2)	0.0559±0.0052	0.0508	0.0611
E_pixel(1)	0.0008	-	-

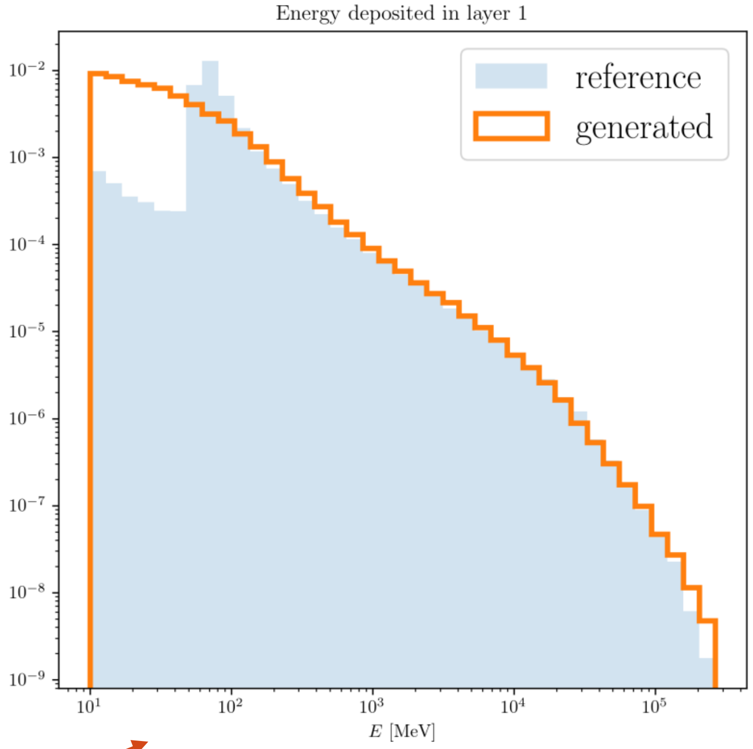

Shower average

Generative Model Performance

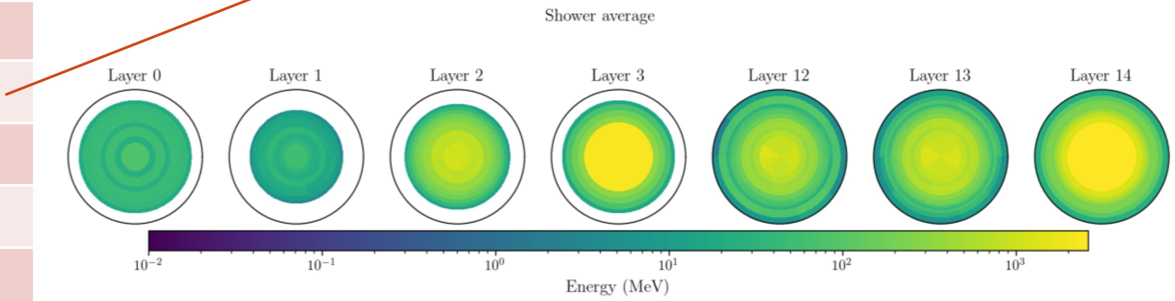
DS1-pion







Scores Cat. (No.):	Mean	Best	Worst
Etot/Einc(16)	0.0081±0.0087	0.0012	0.0365
Esum(7)	0.0580±0.0626	0.0033	0.1800
Eta_Center(4)	0.0386±0.0420	0.0070	0.1100
Phi_Center(4)	0.0363±0.0294	0.0122	0.0865
Eta_Width(4)	0.0801±0.0655	0.0158	0.1785
Phi_Width(4)	0.0796±0.0673	0.0148	0.1791
E_pixel(1)	0.0020	-	-

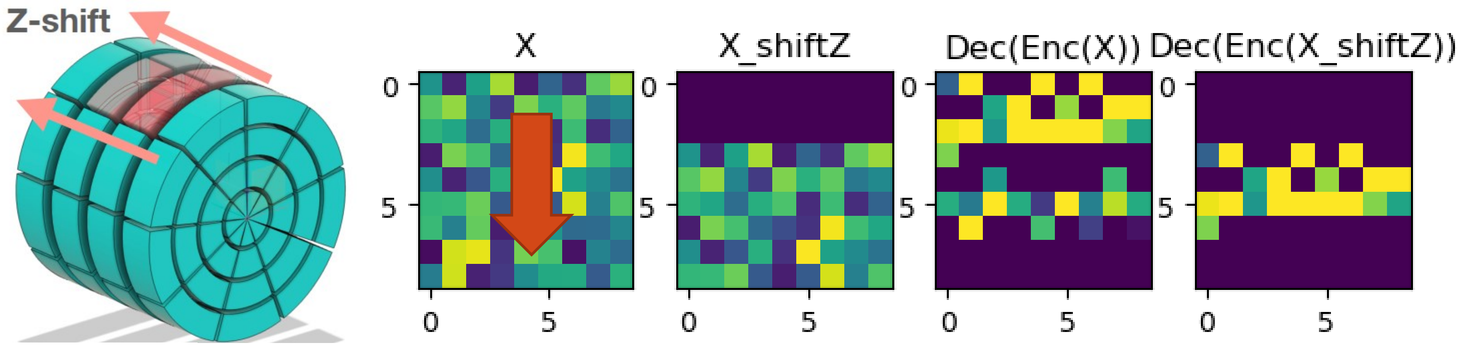


ds2&3

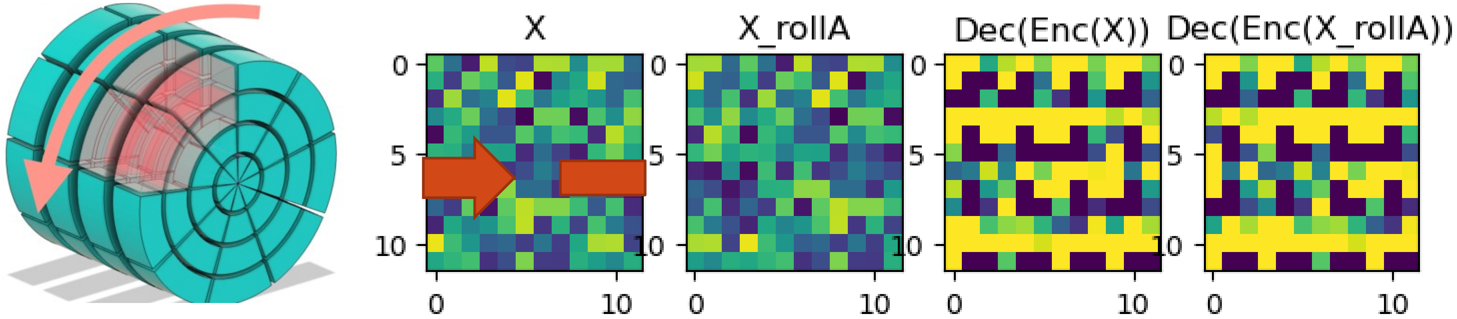
Dataset2&3

Cylindrical Convolution for Special Geometry

- Custom conv operator defined for the geometry
  - 2D kernel and convolution in Z and Alpha direction (R direction treated as channel)
  - Circular boundary for the Alpha direction
  - Standard convolution in Z direction
- The equivariant with translation kept



Alpha-shift



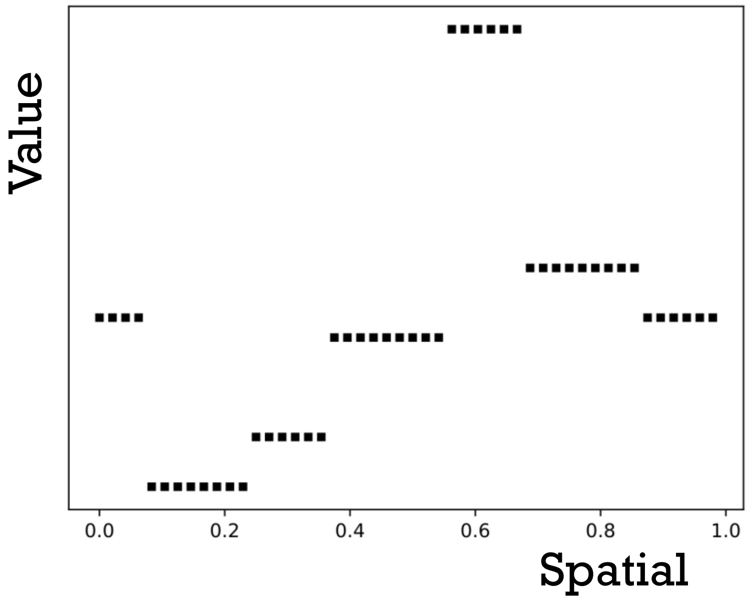
FFT resampling

- How to proper handle 50 circular segmentations: Cyclic stride only possible w/ integer divisor
- FFT resampling! Flexibility for any arbitrary circular dimension
- Replacement of simple stride in convolution operation for up/down-sampling

Demo of FFT resample (1D)

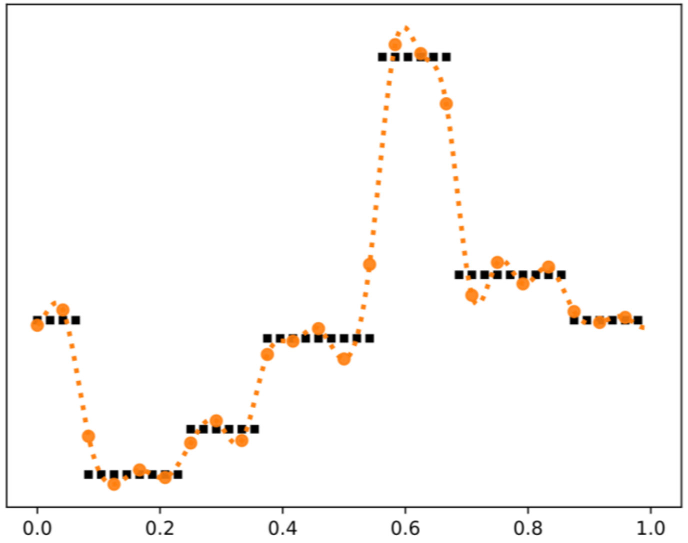
Cyclic data (e.g. 1D L=50)

```
x <- data
```



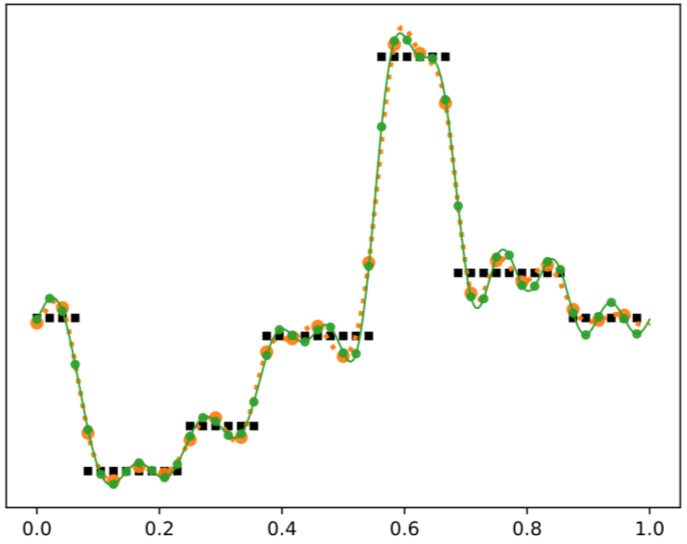
FFT Downsampling → L=26

```
x_down = ifft(
    rfft(x)[:-50//2])
```



FFT Upsampling → L=50

```
x_up = ifft(
    pad(rfft(x_down),(0,N//2)))
```

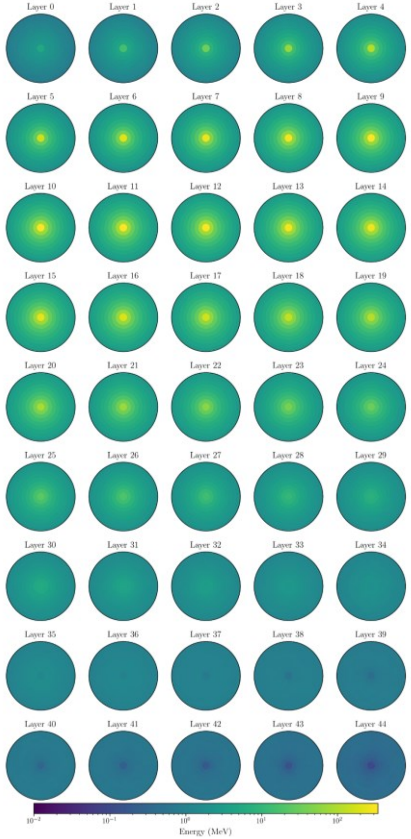


Generative Performance (Average and Individual Image)

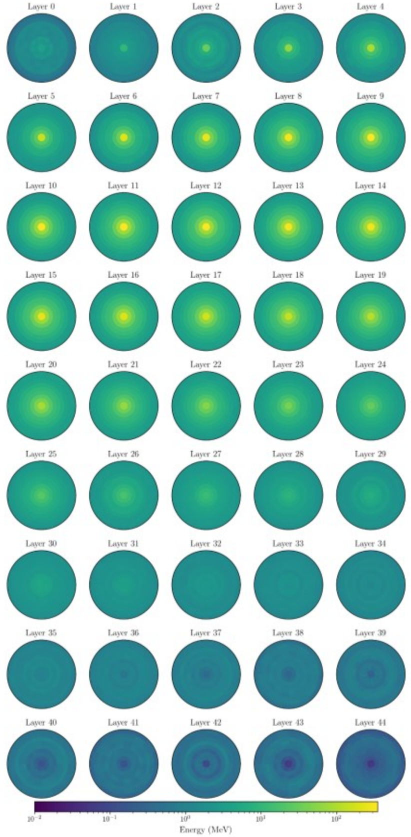
DS2

- Average image reflects the energy distribution across each layer and each "ring"
- Individual image shows the randomness of sampling and the underlying physics (shower)
- Current model gives good performance for the two aspects

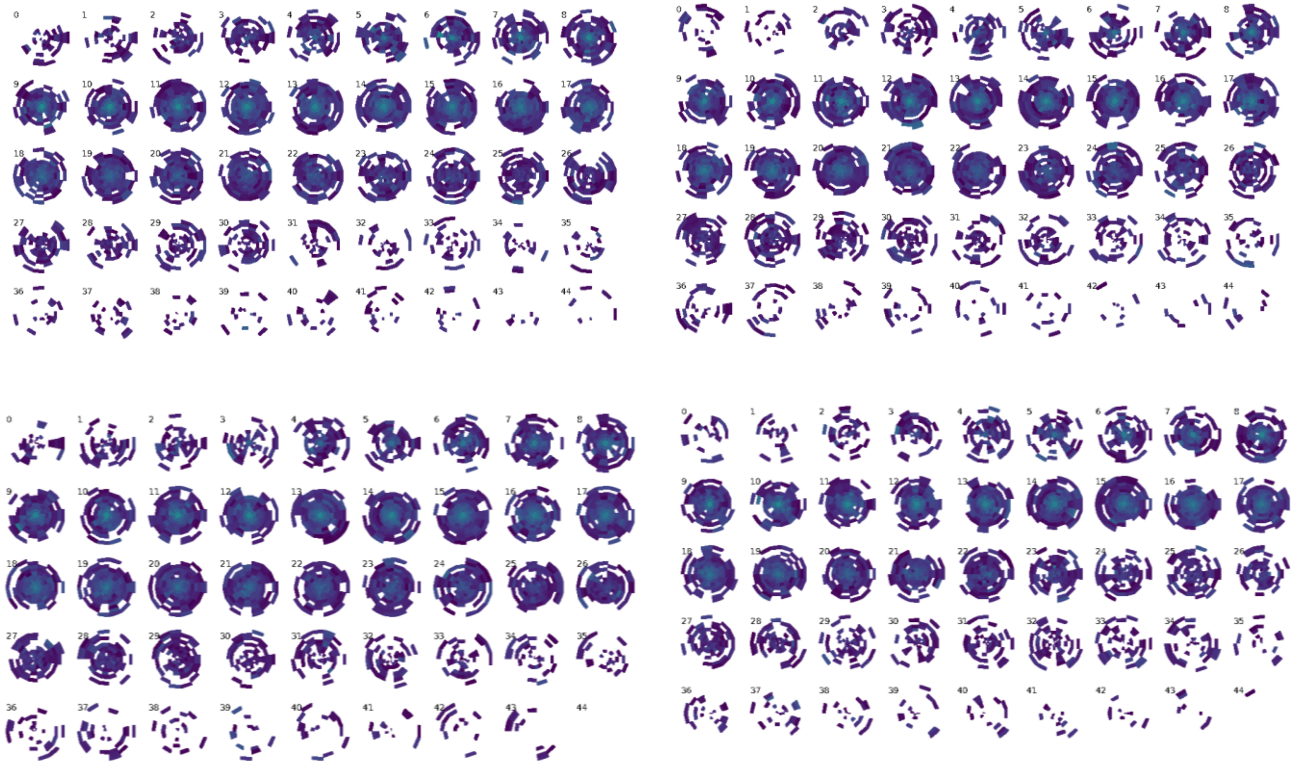
Truth Avg.



Sampled Avg.



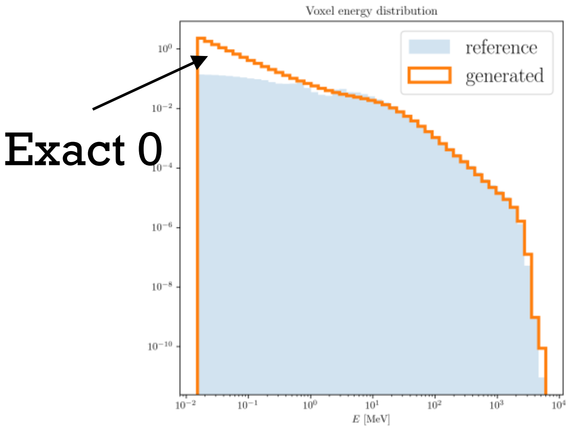
E_inc=745.11GeV (4 samplings)

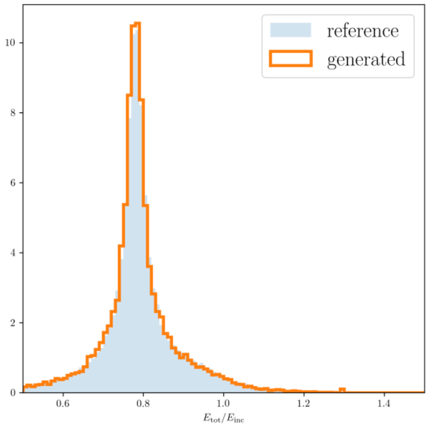


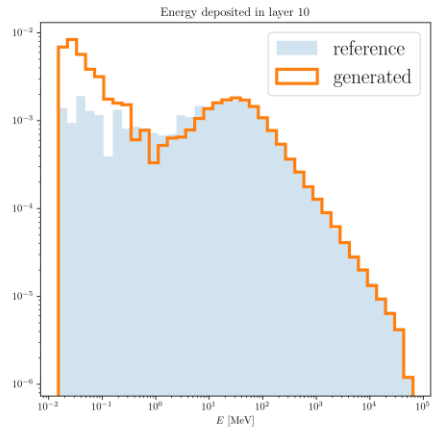
Generative Performance (Distribution)

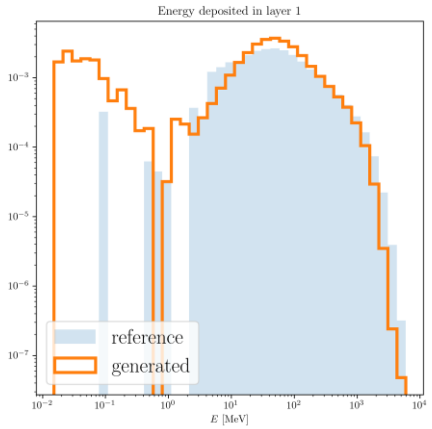
DS2

- The official calo challenge metrics evaluated and we report based on different category:
  - Pixel energy, total energy, R and energy weighted center&width
  - For each category, the metrics are averaged and the best layer and worst layer are listed for reference

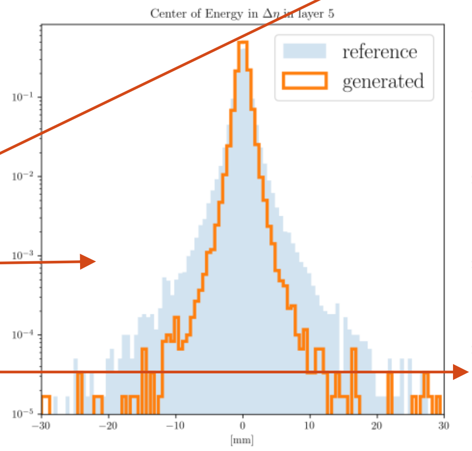


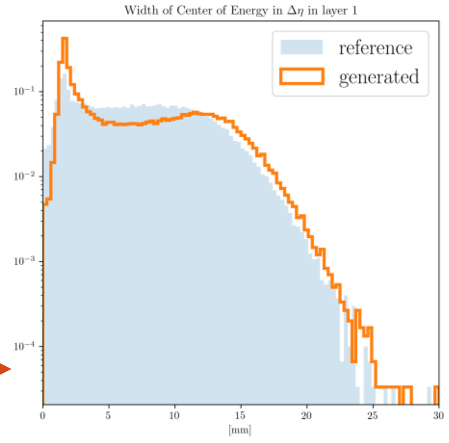






Scores Cat. (No.):	Mean	Best Layer	Worst Layer
E_pixel(1)	0.0506	-	-
Etot/Einc(1)	0.0022	-	-
E_sum(45)	0.0080±0.0058	0.0007	0.0259
Eta_Center(45)	0.0100±0.0082	0.0016	0.0282
Phi_Center(45)	0.0101±0.0082	0.0023	0.0293
Eta_Width(45)	0.0192±0.0126	0.0032	0.0512
Phi_Width(45)	0.0190±0.0125	0.0033	0.0508



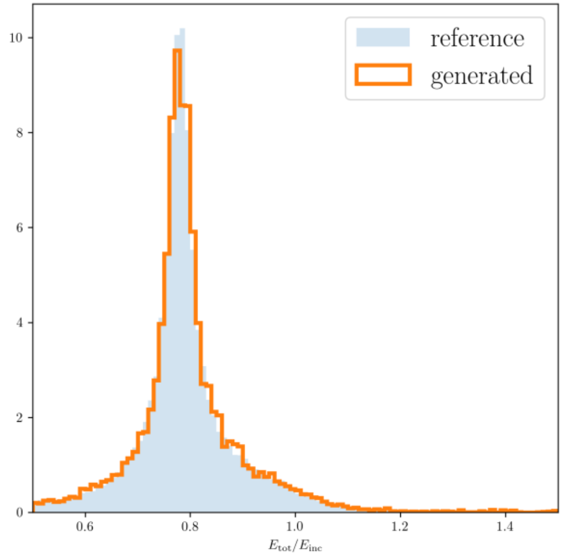


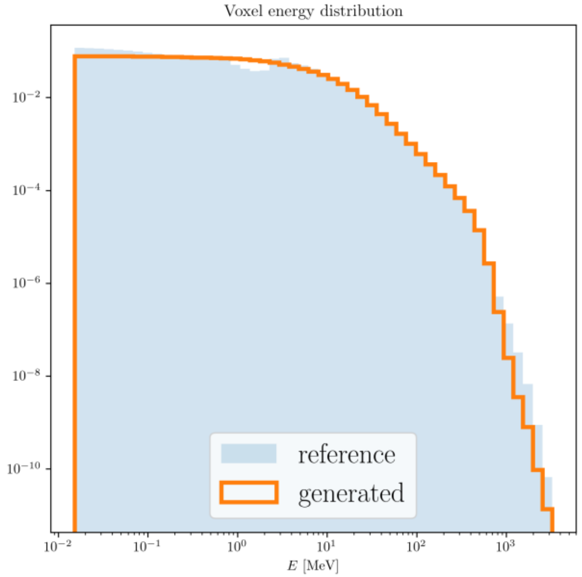
Generative Performance (Distribution)

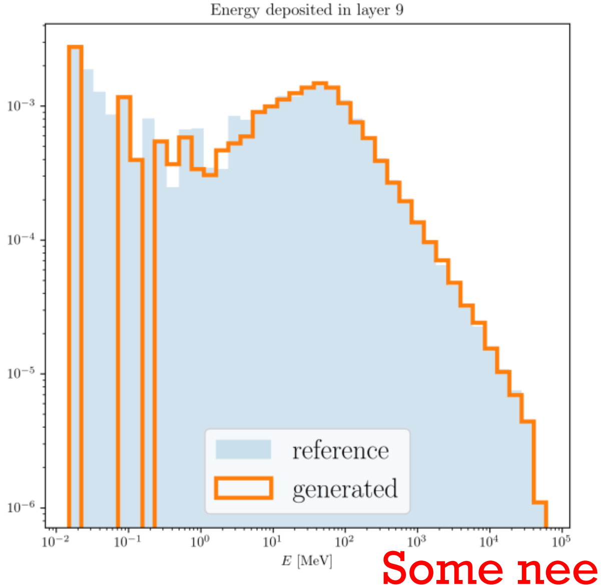
DS3

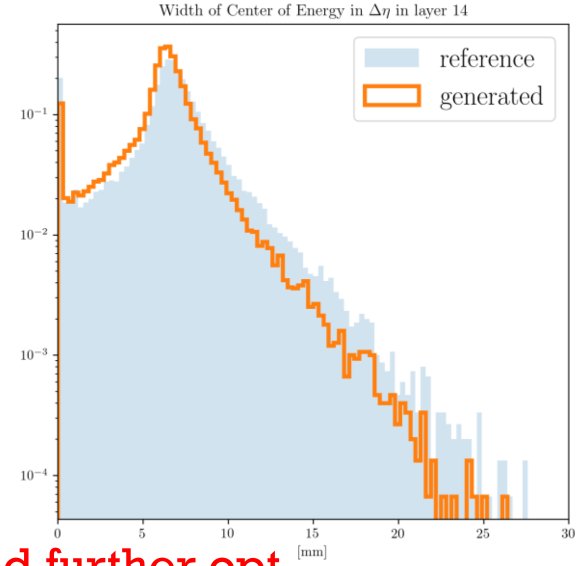
- Performance of ds3 model: generally fine but some layers need further improvement

Some good ones



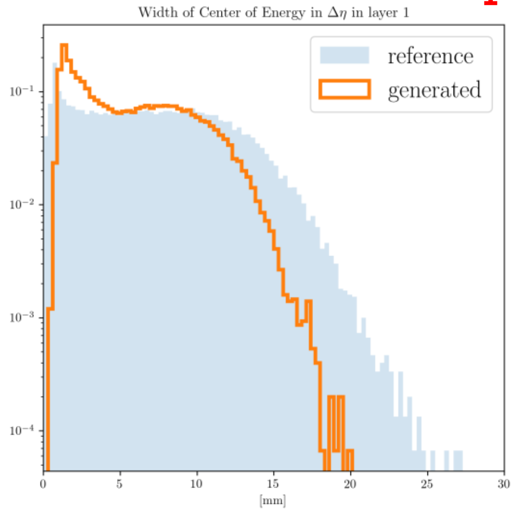

Voxel energy distribution


Energy deposited in layer 9


Width of Center of Energy in $\Delta\eta$ in layer 14

Scores Cat. (No.):	Mean	Best Layer	Worst Layer
Etot/Einc(1)	0.0037	-	-
Esum(45)	0.0203±0.0239	0.0008	0.1276
Eta_Center(45)	0.0245±0.0206	0.0036	0.0839
Phi_Center(45)	0.0246±0.0207	0.0038	0.0841
Eta_Width(45)	0.0426±0.0307	0.0062	0.1359
Phi_Width(45)	0.0424±0.0295	0.0062	0.1264
E_pixel(1)	0.0078	-	-

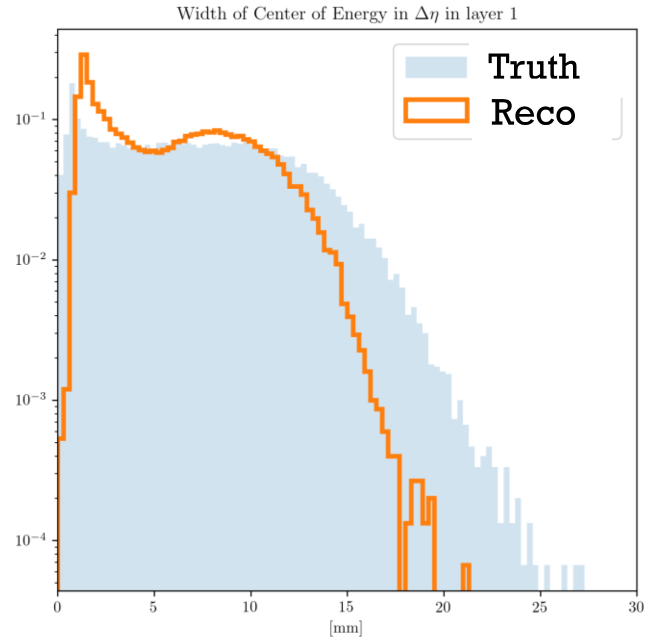
Some need further opt


Width of Center of Energy in $\Delta\eta$ in layer 1

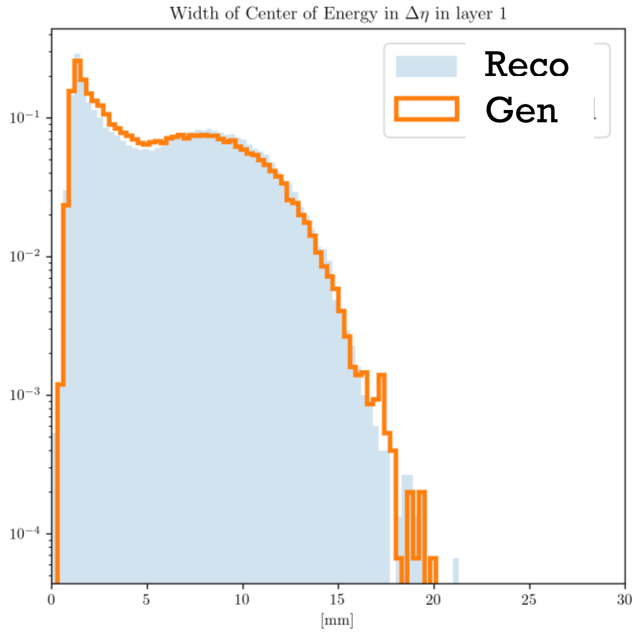
Generated v.s. Reconstructed

- As one unique feature, the two stages generative model could factorize the model and explicitly evaluate each step:
  - "Reconstruction": forward pass of the AE architecture which involves no latent sampling
    - Faithfully recover the truth
  - "Generation": latent sampling + decoder-only which is the real generative mode
    - Sampling the consistent discrete joint distribution in latent space which is much compressed
- GPT-based latent model (Stage2) does perfect job and the problem reduce to just tune an AE

Truth v.s. Reconstruction



Reconstruction v.s. Sampling



Timing and Parameters of models

DS1&2&3

- Parameters count all the trainable parameters of the model
  - At several M level (S1 + S2 : 4M+0.2M for ds1, 3.1M+3.7M for ds2, 2.1M+0.4K for ds3)

not fully optimized due to time constraint

- Timing measured with V1001 32GB
- Most time spent in latent sampling and depend on the size of latent space
  - Latent model (GPT) could be optimized smaller for better speedup with large batch size
  - S1 compression time is negligible and allowing for combining with other techniques
  - A lot of further optimizations for S1 and won't slow down the generation

Datasets (sampling)	S1 time/ms	S2 time/ms	Total time/ms
ds1-photon	0.017	0.081	0.098
ds1-pion	0.017	0.084	0.101
ds2	0.140	1.790	1.930
ds3	0.208	0.695	0.903

VI

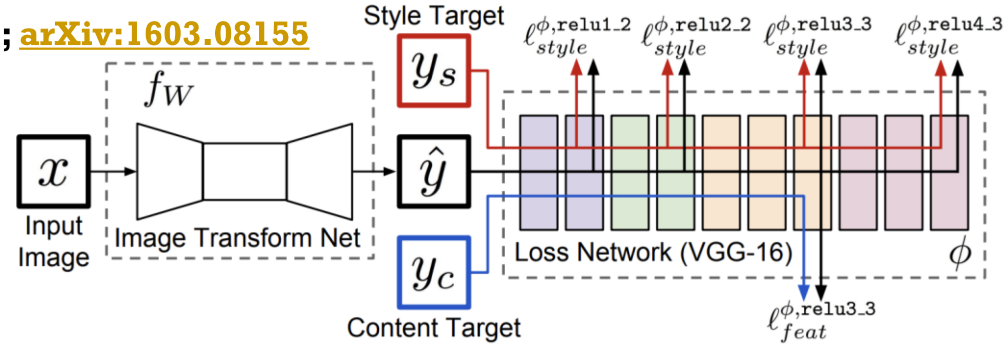
More Ideas

More “meaningful losss” for VAE?

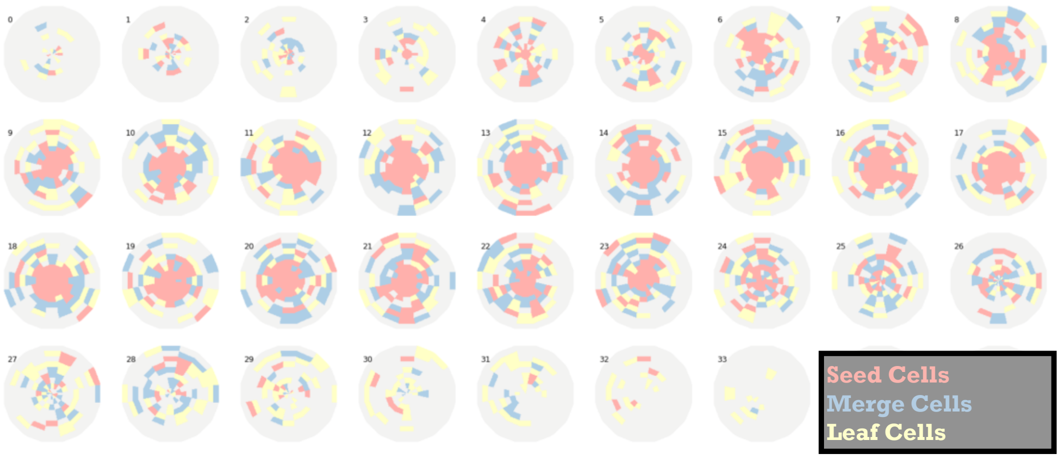
Physics motivated Perceptual Loss

- Pre-trained network to match high-level features
- More tolerant than per-pixel loss to perceptually (or physically) irrelevant differences
- Pretrained network suitable for this dataset?
  - ➞ We can invent a task to solve!
  - ➞ Should be nontrivial, require the network to learn physically-relevant features

Johnson et al; arXiv:1603.08155


Style Target
y_s
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
f_W
x
Input Image
Image Transform Net
$\hat{y}$
y_c
Content Target
Loss Network (VGG-16)
ϕ
$\ell_{feat}^{\phi,relu3\_3}$

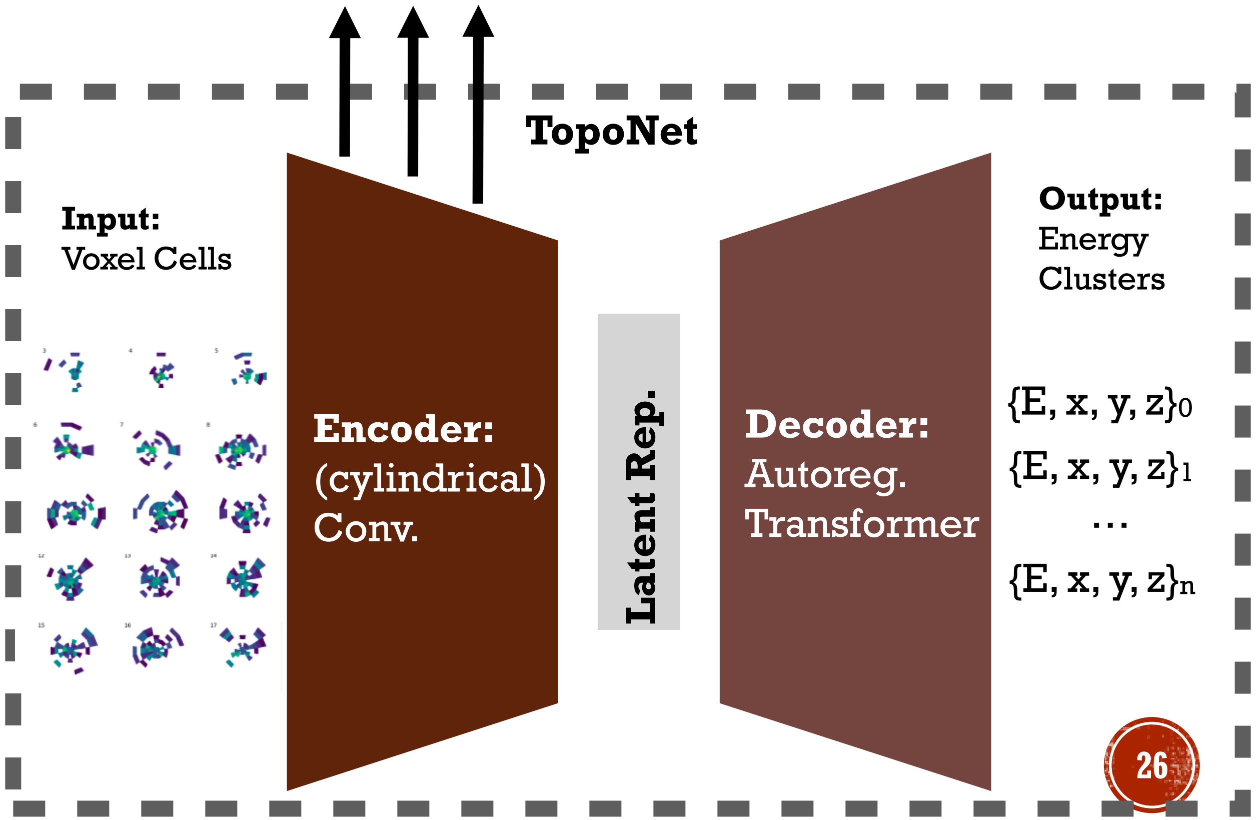
Proxy Task: Topo Clustering on DS2



Calculate topoclusters directly per event (4-2-0-ish)

Truth labels ➞ **for training**

Hidden layers used for calo feature losses



Conclusion and Future Plan

- Calorimeter simulation with vital importance in HEP but challenging in computing
- Machine learning methods show great potential for calo sim speedup
- Two stages model proposed based on VAE-like architecture
  - Vector quantization for good compression and flexible latent space
  - GPT model adapted to conditional sample in the latent space
- Methods dealing with special dataset(s) presented
  - Energy normalization by condition E and total E
  - 2D cylindrical conv designed to deal with the cylindrical boundary condition
  - FFT resampling to up/down-sample for arbitrary cylindrical geometry
- Results presented for dataset1,2,3 with promising results
- Performance such as classifier score limited by Stage1 and could be further improved
- More ideas discussed and in hope of the physics-aware calo sim model in the future

Thanks to your listening!

Backups

Preprocessing and Conditioning

DS1

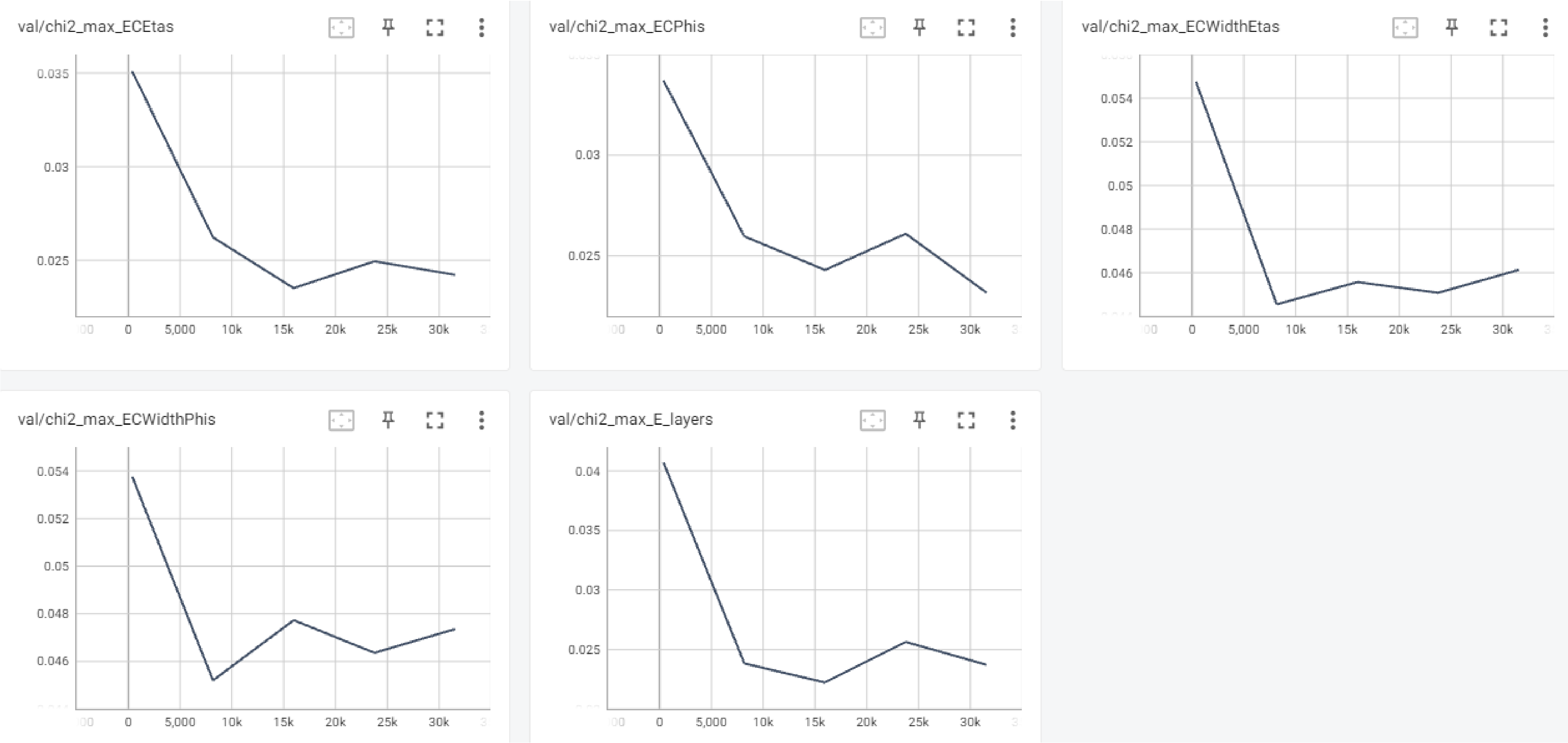
- Simple flatten to handle the irregular geometry → simple MLP dense layers and conv1D used
- Binned condition (one hot embedding) used in S1 and S2

DS2&3

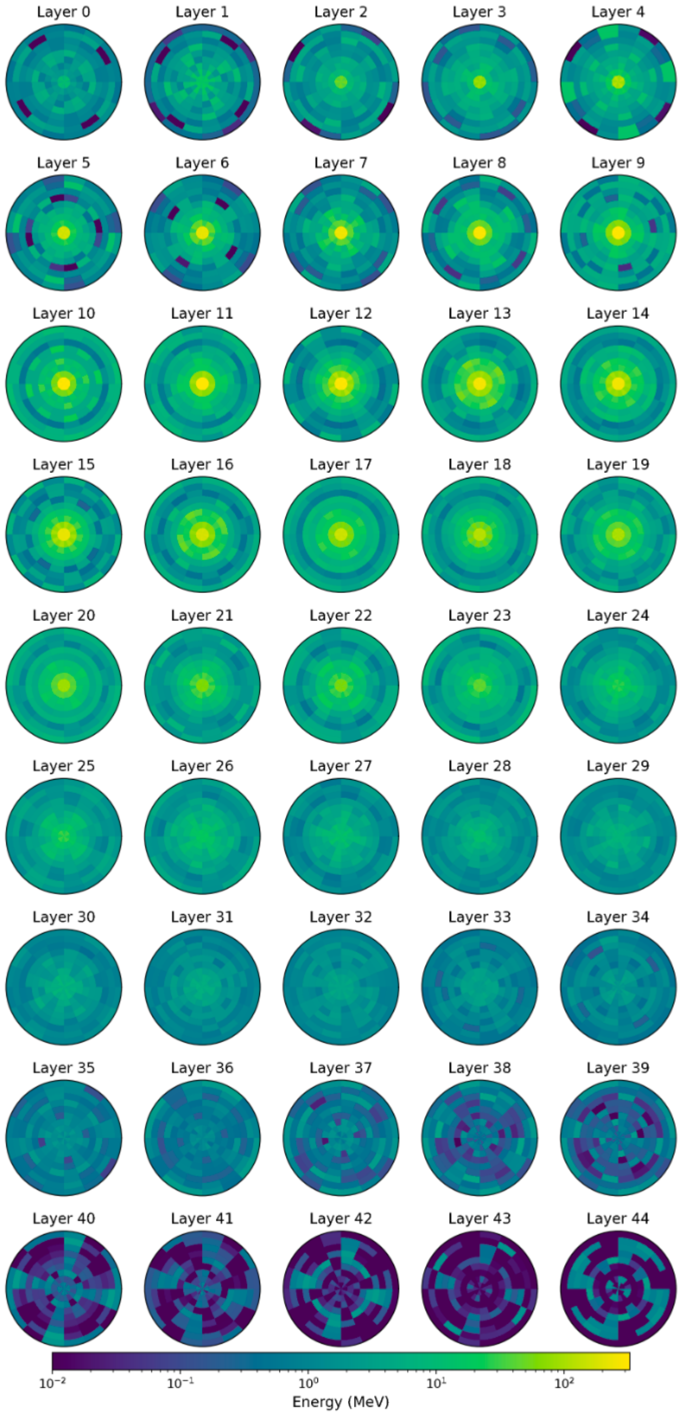
- Special cylindrical geometry with nontrivial translative symmetry:
  - In Z direction: standard translation symmetry
  - In Alpha direction: translation symmetry with circular boundary
  - **In R direction: NO translation symmetry**
- To save the complexity of 3D-conv operation and utilize the special symmetry:
  - Geometry remapped to (Z * A) * R where R is treated as channel of the 2D `Z*A` image
  - Custom cylindrical convolution and FFT resampling to implement the conv network
- Condition takes logarithm and remap to [-1,1] then append as extra channel in S1
- Condition with same processing used in S2 model

Training and Optimization/model picking

- Optimizer uses common Adam and LR tuned (starts from the maximum LR not collapse)
- Monitor the metrics on-the-fly and pick model (after loss converged) with best



non FFT conv for dataset3 (not work)


Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
Layer 28
Layer 29
Layer 30
Layer 31
Layer 32
Layer 33
Layer 34
Layer 35
Layer 36
Layer 37
Layer 38
Layer 39
Layer 40
Layer 41
Layer 42
Layer 43
Layer 44
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Energy (MeV)