Generative Model for CEPC Detector Simulation

- Review of Previous Work
- Unified Approach for Training and Validation
- Update of Loss Function
- The Geometry of the Detector
- Future Work

Review of Previous Work

- Generation Strategy

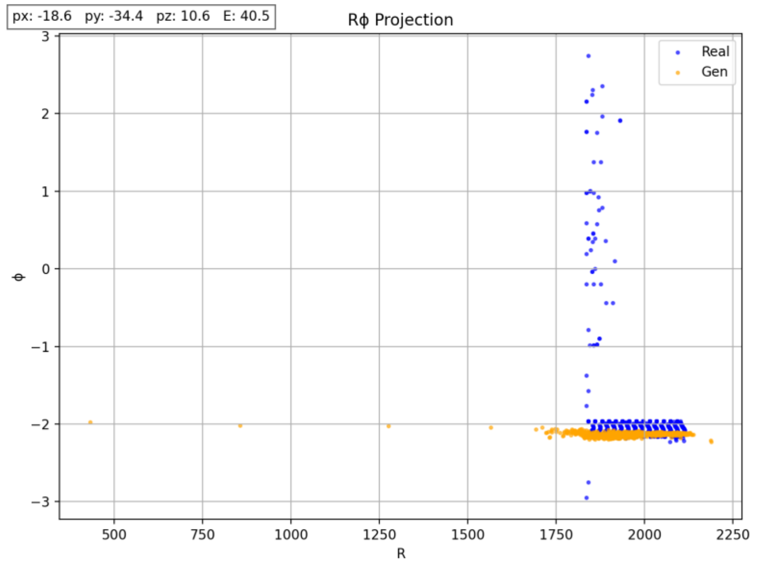
$$p_4 \rightarrow [cell\ 1, cell\ 2, cell\ 3, \ldots, cell\ n]$$

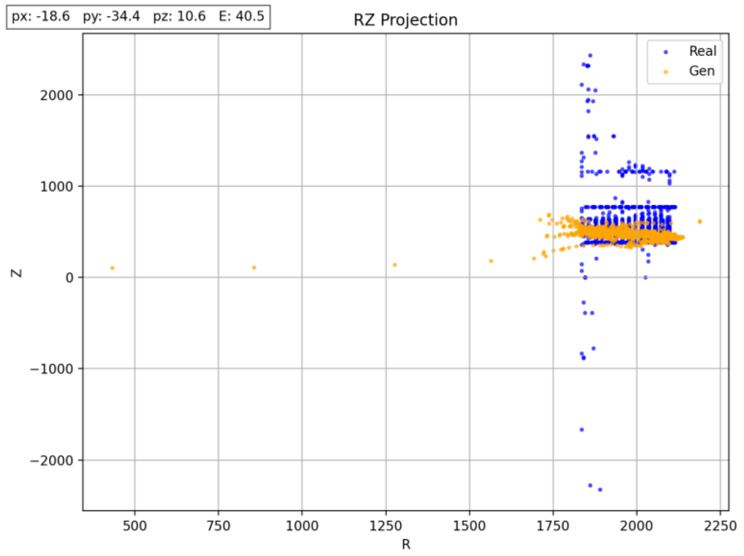
Step	Input	Prediction
1	BOS	Cell 1
2	BOS, Cell 1	Cell 2
...	...	...
n	BOS, cell 1 ... cell n-1	Cell n-1

Our generation strategy is somewhat unconventional. We arrange the cell information into a sequence, sorted by energy from highest to lowest, and then let the model generate this sequence step by step.
Since the number of cells varies depending on the input four-momentum, we need a model capable of handling variable-length sequence generation. We initially experimented with WGAN-GP, but GAN generally require fixed-dimensional outputs. Therefore, we switched to using a Transformer-based approach instead.

Review of Previous Work

- Previous Result





Total Energy : Real: 37.0530, Gen: 43.6340

In our previous results, the generated clusters were already well reconstructed in terms of their positions. However, the fine details of their shapes still require improvement. This suggests that incorporating the detector's geometric information into the model is necessary.

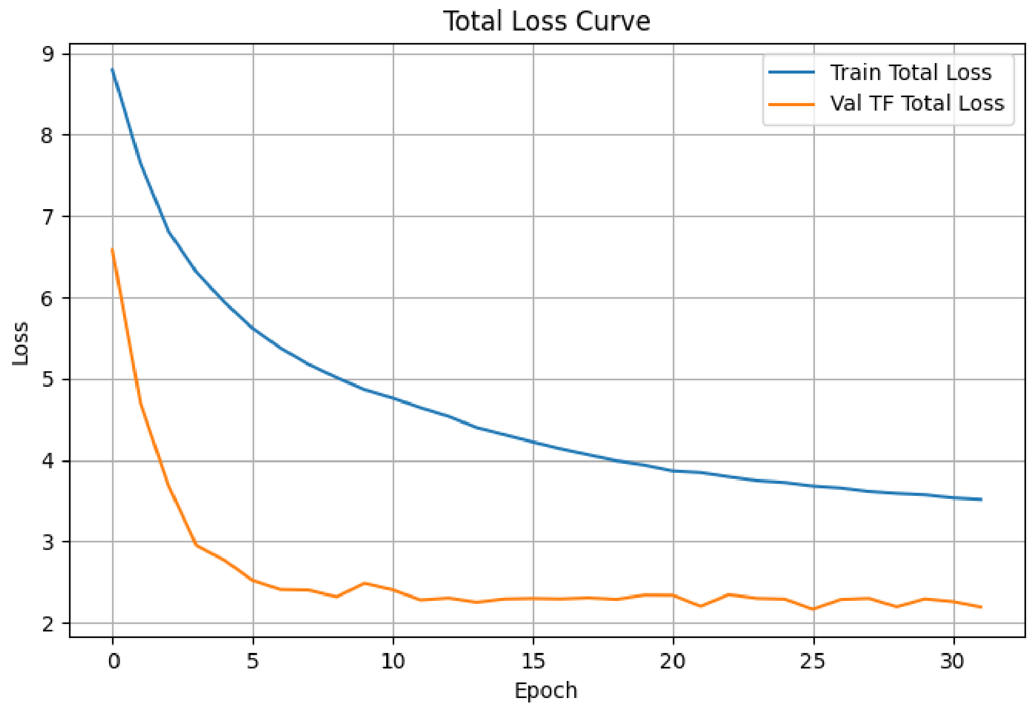
In addition, the model has not yet learned well when to stop the generation process. As a result, the total energy of the generated cluster tends to be higher than the original input energy.

Unified Approach for Training and Validation

In previous training runs, the validation loss appeared significantly better than the training loss. This was due to an inconsistency in the loss calculation strategies between the two phases.

We have now corrected this issue by aligning the loss computation approach. Specifically, the **training loss** is calculated by running the updated model (after the current epoch's parameter updates) on the entire training set in a forward pass.

Similarly, the **validation loss** is computed by applying the same model to the entire validation set in a forward pass.

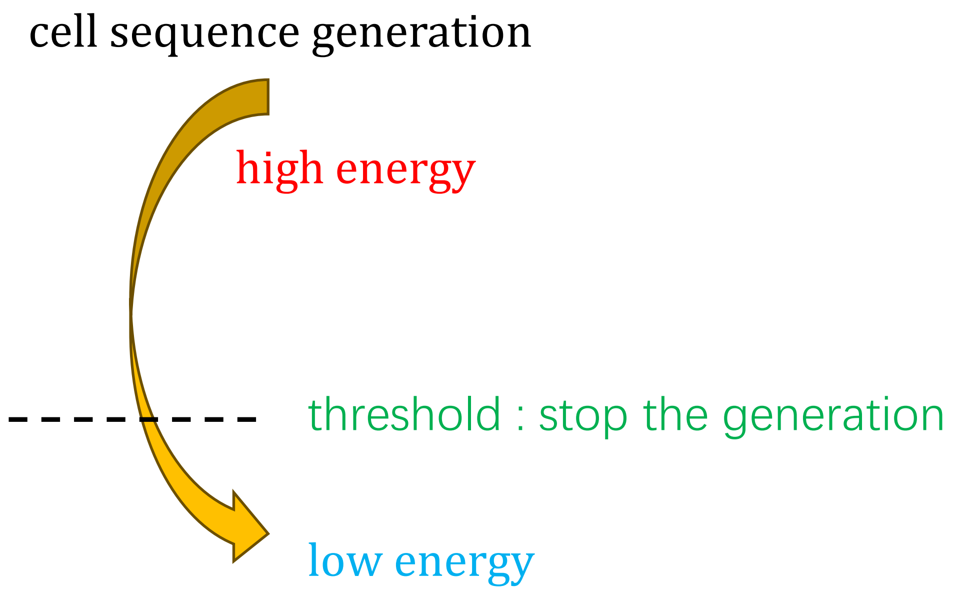


Update of Loss Function

- **Loss of JS(x,y,z,energy)**: Measures the similarity between prediction and target.
  We corrected an issue where the JS divergence was calculated using hard binning, which prevented gradients from being properly backpropagated. To address this, we switched to using a soft histogram approach.

$$\text{js\_loss} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{4}\sum_{d=1}^{4}\text{JS}(\text{hist\_pred}_{b,d}, \text{hist\_target}_{b,d})$$

- **Loss of EOS**: We decided to abandon the EOS (End-of-Sequence) loss due to its poor performance. Instead, we introduced an energy threshold to determine when the model should stop generating. However, this strategy requires the model to learn the energy-based ordering accurately.

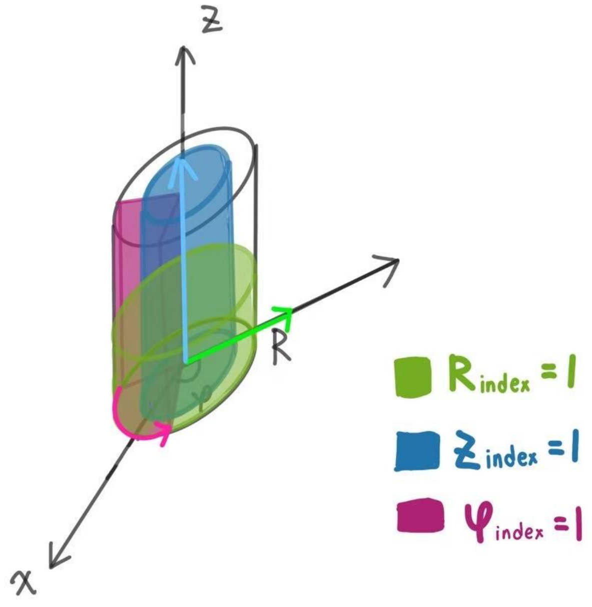


The Geometry of the Detector

To help the model learn the detector's geometric structure, we implemented a discretization (gridding) of the detector space. Specifically, we first convert the cell positions from Cartesian coordinates to cylindrical coordinates. Then, we divide the detector's geometric range into bins along the R, Z, φ directions. The cylindrical coordinate values are mapped to their corresponding bin indices, effectively transforming the continuous positions into integer-encoded representations.

ECAL BARREL

parameter	value
Inner radius	1830 mm
Outer radius	2130 mm
Thickness	300 mm
half length	2900 mm
gap between	30 mm
Layer	28
number of phi modules	32
number of z modules	15



$$
\begin{gathered}
(x, y, z) \\
\downarrow \\
(R, Z, \varphi) \\
\downarrow \\
(index\_R, index\_Z, index\_\varphi)
\end{gathered}
$$

The Geometry of the Detector

Some issues need to be considered

- The rounding operation cannot be used as part of the activation function, as it would block gradient flow during training. Therefore, rounding is only applied during inference, while floating-point values are maintained during training.

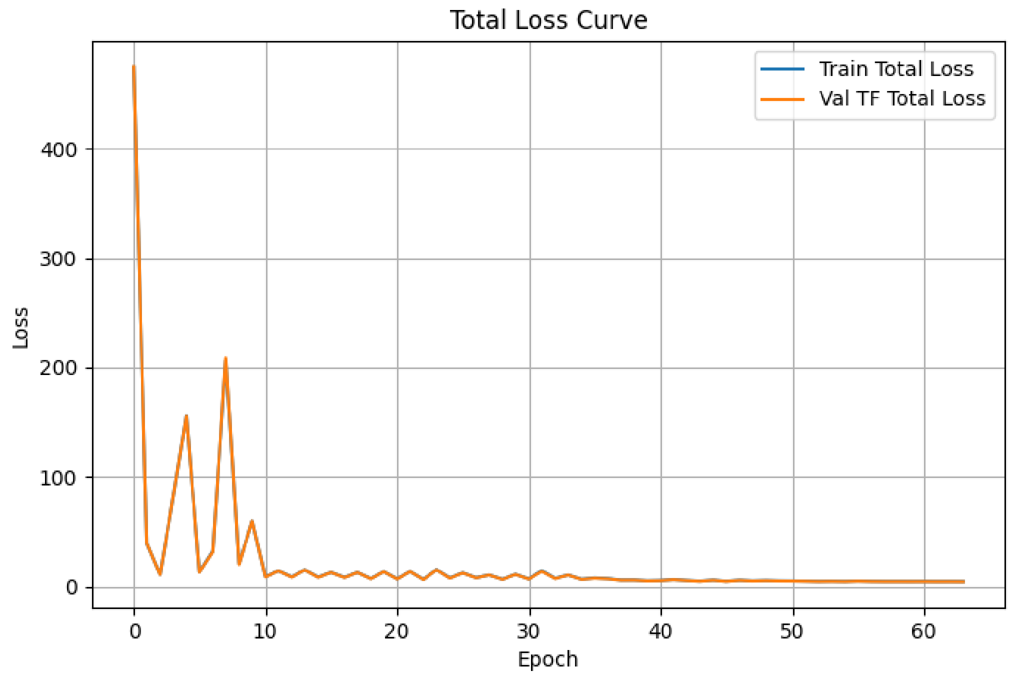
- After converting coordinates into indices, the positional values become relatively large, which may overshadow the energy feature. To address this, we assign a separate weight to the energy term in the loss function, effectively amplifying the energy contribution to balance it against the positional features.

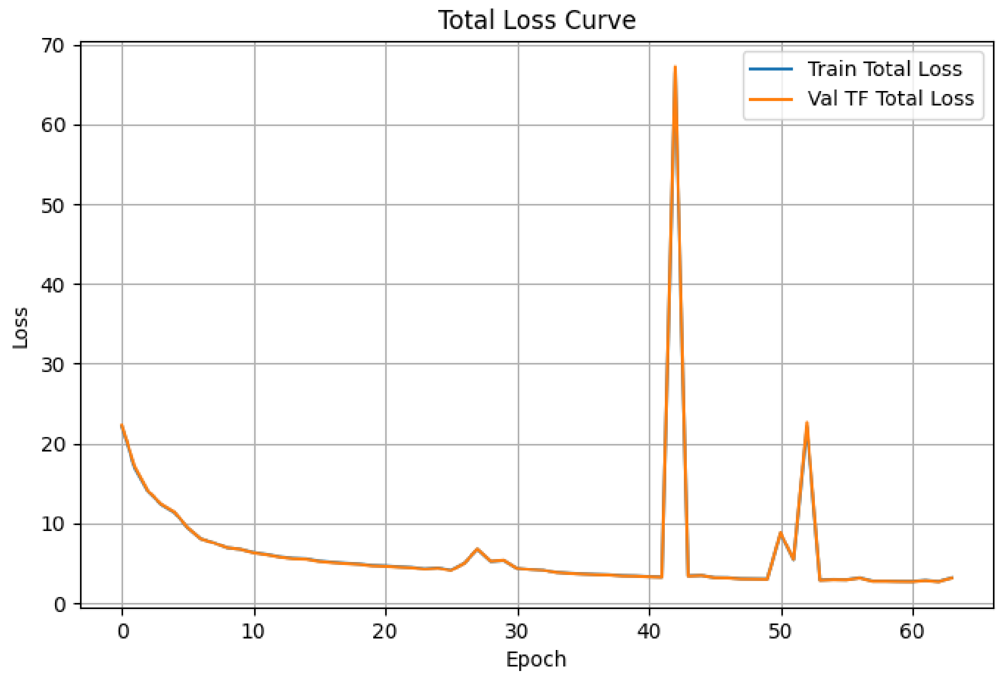
- The current binning resolution of (28,15,32)may be too rough. Many cells are mapped to the same bin, leading to significant loss of spatial information in the clusters.

The Geometry of the Detector

Training Results

The overall loss is decreasing as expected, but the total loss still shows significant fluctuations, mainly driven by variations in the **Total Energy Loss** term.



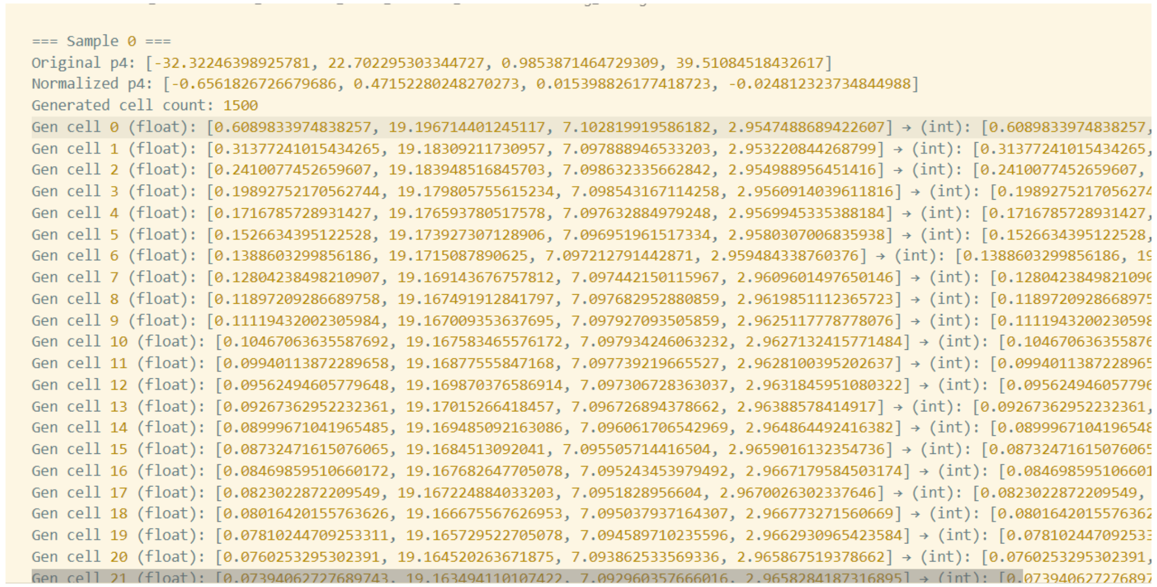


$$Epoch = 64, Batch_{size} = 64, Num_{head} = 8, Num_{layer} = 4$$

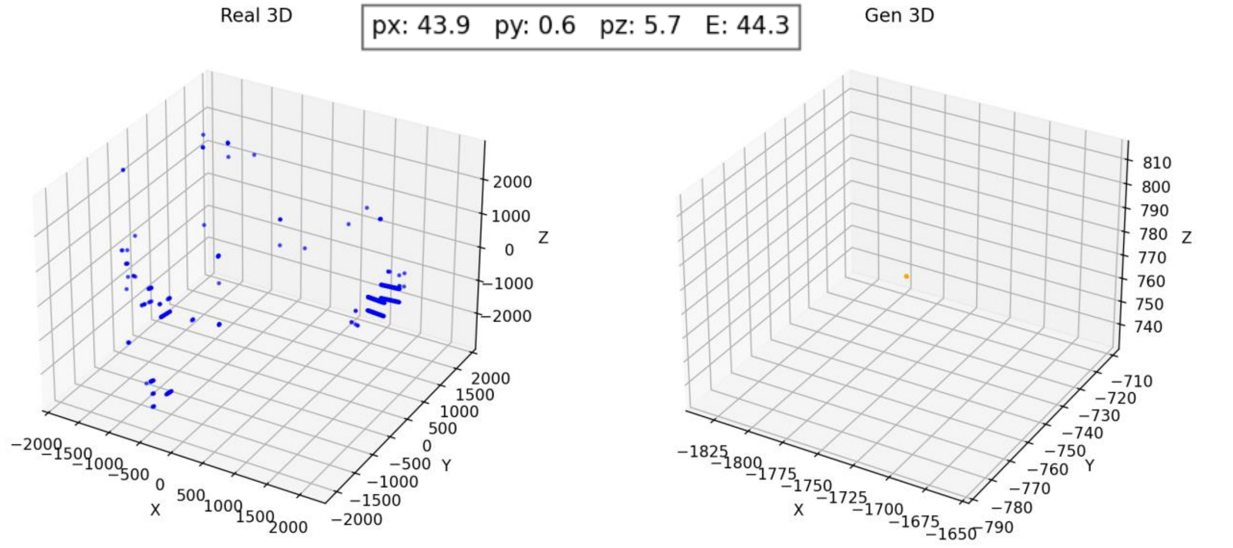
The Geometry of the Detector

Current Issue — Mode Collapse in Output

When using the current trained model for generation, we observed that the model tends to produce cells concentrated around a single location. After applying rounding, the entire generated cell sequence becomes identical. This suggests that the model has not yet learned to generate diverse or realistic spatial distributions.



```
=== Sample 0 ===
Original p4: [-32.32246398925781, 22.702295303344727, 0.9853871464729309, 39.51084518432617]
Normalized p4: [-0.6561826726679686, 0.4715228024827027З, 0.015398826177418723, -0.024812323734844988]
Generated cell count: 1500
Gen cell 0 (float): [0.6089833974838257, 19.196714401245117, 7.102819919586182, 2.9547488689422607] → (int): [0.6089833974838257,
Gen cell 1 (float): [0.31377241015434265, 19.18309211730957, 7.097888946533203, 2.953220844268799] → (int): [0.31377241015434265,
Gen cell 2 (float): [0.241007745265960‍7, 19.183948516845703, 7.098632335662842, 2.954988956451416] → (int): [0.2410077452659607,
Gen cell 3 (float): [0.19892752170562744, 19.179805755615234, 7.098543167114258, 2.9560914039611816] → (int): [0.1989275217056274
Gen cell 4 (float): [0.1716785728931427, 19.176593780517578, 7.097632884979248, 2.9569945335388184] → (int): [0.1716785728931427,
Gen cell 5 (float): [0.1526634395122528, 19.173927307128906, 7.096951961517334, 2.9580307006835938] → (int): [0.1526634395122528,
Gen cell 6 (float): [0.1388603299856186, 19.1715087890625, 7.097212791442871, 2.959484338760376] → (int): [0.1388603299856186, 19
Gen cell 7 (float): [0.12804238498210907, 19.169143676757812, 7.097442150115967, 2.9609601497650146] → (int): [0.1280423849821090
Gen cell 8 (float): [0.11897209286689758, 19.167491912841797, 7.097682952880859, 2.9619851112365723] → (int): [0.1189720928668975
Gen cell 9 (float): [0.11119432002305984, 19.167009353637695, 7.097927093505859, 2.9625117778778076] → (int): [0.1111943200230598
Gen cell 10 (float): [0.10467063635587692, 19.167583465576172, 7.097934246063232, 2.9627132415771484] → (int): [0.1046706363558769
Gen cell 11 (float): [0.09940113872289658, 19.16877555847168, 7.097739219665527, 2.9628100395202637] → (int): [0.0994011387228965
Gen cell 12 (float): [0.09562494605779648, 19.169870376586914, 7.097306728363037, 2.9631845951080322] → (int): [0.0956249460577964
Gen cell 13 (float): [0.09267362952232361, 19.17015266418457, 7.096726894378662, 2.963885784149017] → (int): [0.09267362952232361,
Gen cell 14 (float): [0.08999671041965485, 19.169485092163086, 7.096061706542969, 2.964864492416382] → (int): [0.0899967104196548
Gen cell 15 (float): [0.08732471615076065, 19.168451309204l, 7.095505714416504, 2.965901613235473‍6] → (int): [0.0873247161507606‍5
Gen cell 16 (float): [0.08469859510660172, 19.167682647705078, 7.095243453979492, 2.9667179584503174] → (int): [0.0846985951066017
Gen cell 17 (float): [0.0823022872209549, 19.167224884033203, 7.0951828956604, 2.967002630233764‍6] → (int): [0.0823022872209549,
Gen cell 18 (float): [0.08016420155763626, 19.166675567626953, 7.095037937164307, 2.9667732715606‍69] → (int): [0.0801642015576362
Gen cell 19 (float): [0.07810244709253311, 19.165729522705078, 7.094589710235596, 2.9662930965423584] → (int): [0.0781024470925331
Gen cell 20 (float): [0.0760253295302391, 19.1645202636718‍75, 7.093862533569336, 2.9658675193786‍62] → (int): [0.0760253295302391,
Gen cell 21 (float): [0.07394062727689743, 19.163494110107422, 7.092960357666016, 2.965828418731689‍5] → (int): [0.0739406272768974
```



The Geometry of the Detector

Some possible reason for the collapse in output

- One possible cause is an inappropriate loss function design. However, we have already tried various combinations of loss functions during training, but the model still produces bland outputs.

- Another possible reason is that the model complexity is too high, while the data representation is too simple. Specifically, our current coarse binning may have lost important structural patterns in the data. As a result, the model might converge to generating a trivial, fixed position that minimizes the loss without truly learning the underlying distribution.

- The current inference process uses autoregressive generation without any perturbation, which makes it prone to producing bland outputs.

Future Work

- Try reducing model complexity.
- Explore finer binning to help the model learn more detailed structures.
- Introduce noise sampling into the regression process.
- Improve the training strategy by adding partial autoregression during teacher forcing to enhance inference performance.

Thank you for listening