中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

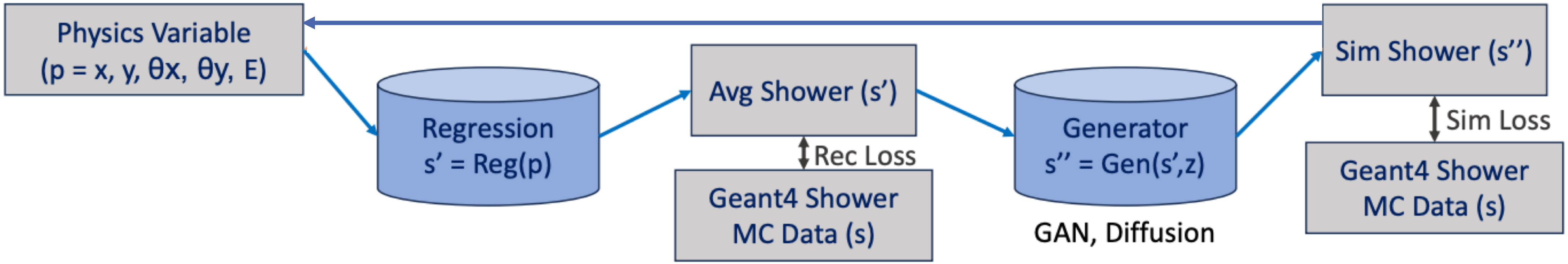
Status report

2025/08/07

ZDC Internal

WAI YUEN CHAN

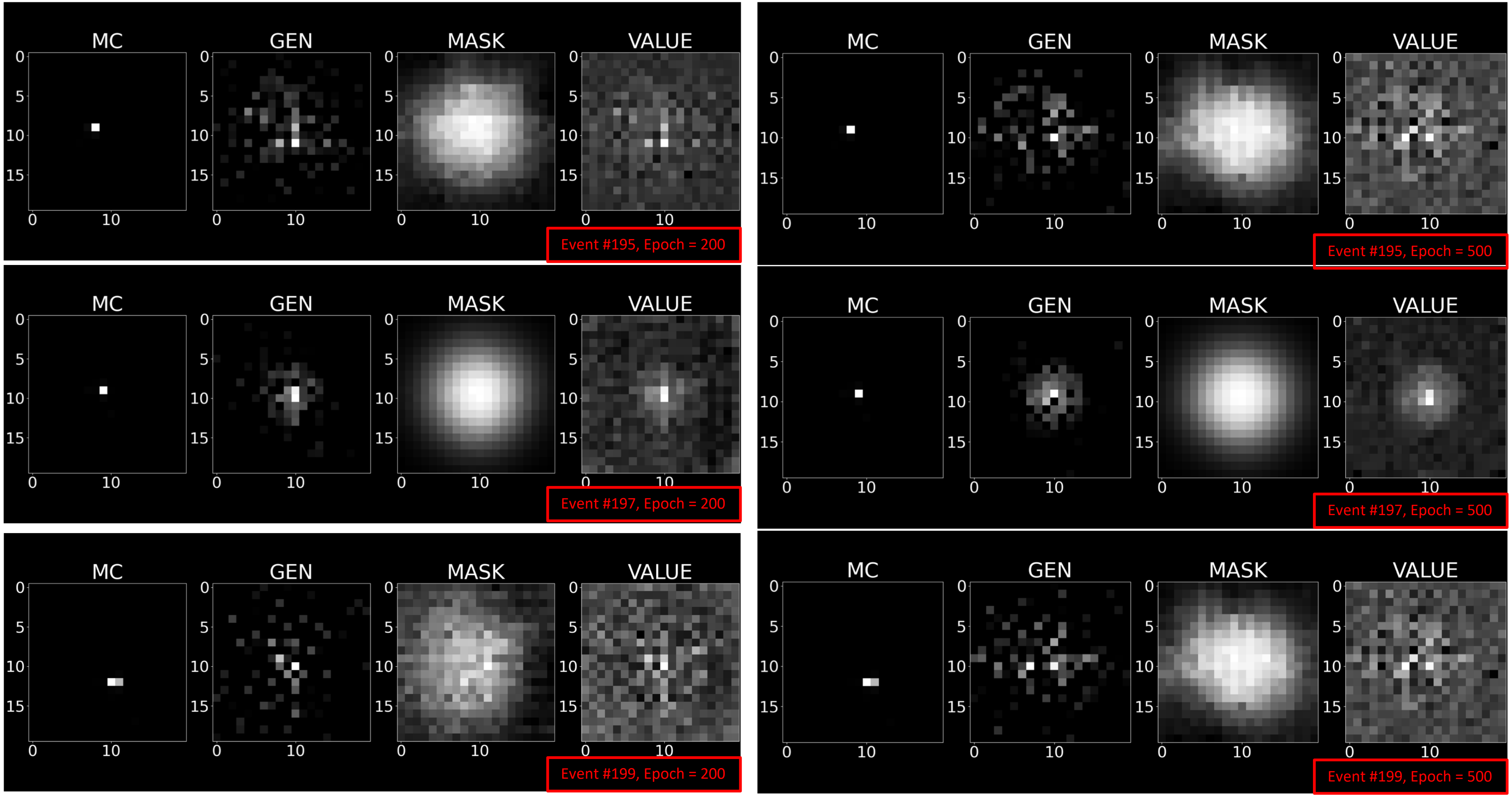
Introduction



We have 2 parts in the algorithm:

1. Regression
   - We train to model to learn the general showering pattern in ECAL
2. Generator
   - We generate fake samples with noise to push the model to learn more details about the pattern (GAN)

Recap: Testing results

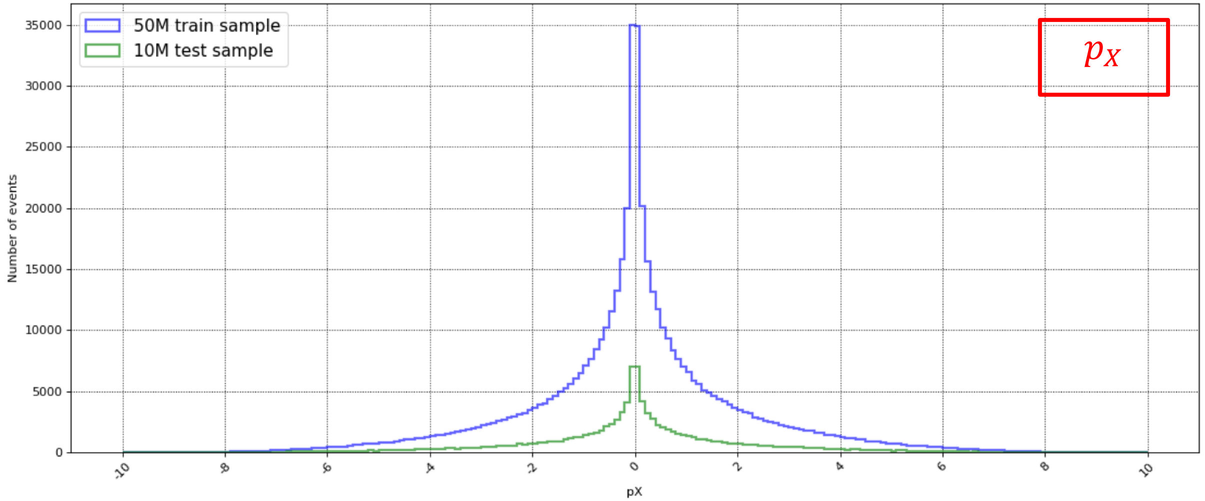


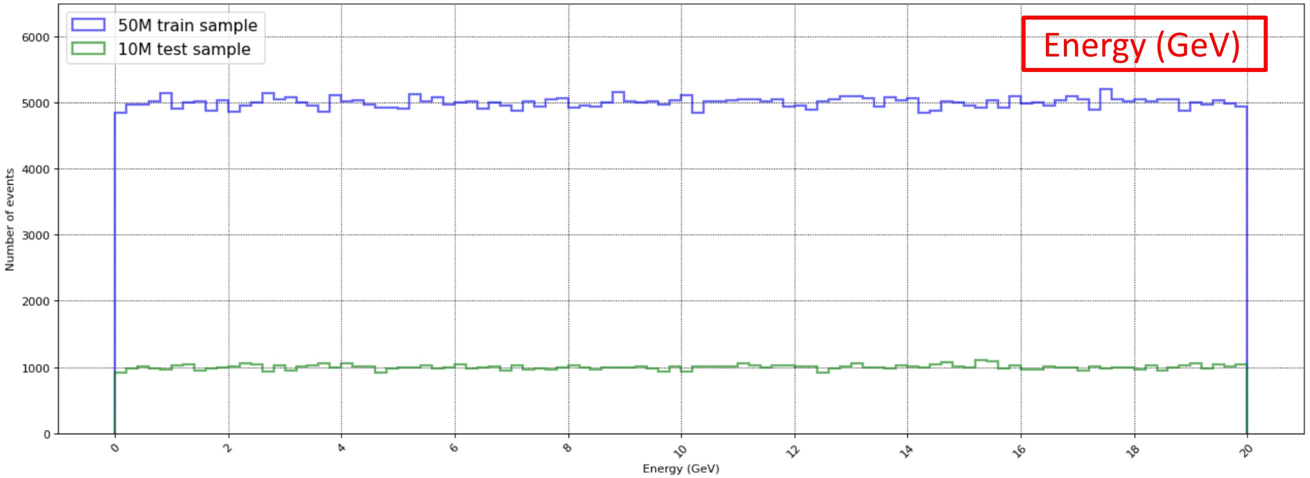
New developments

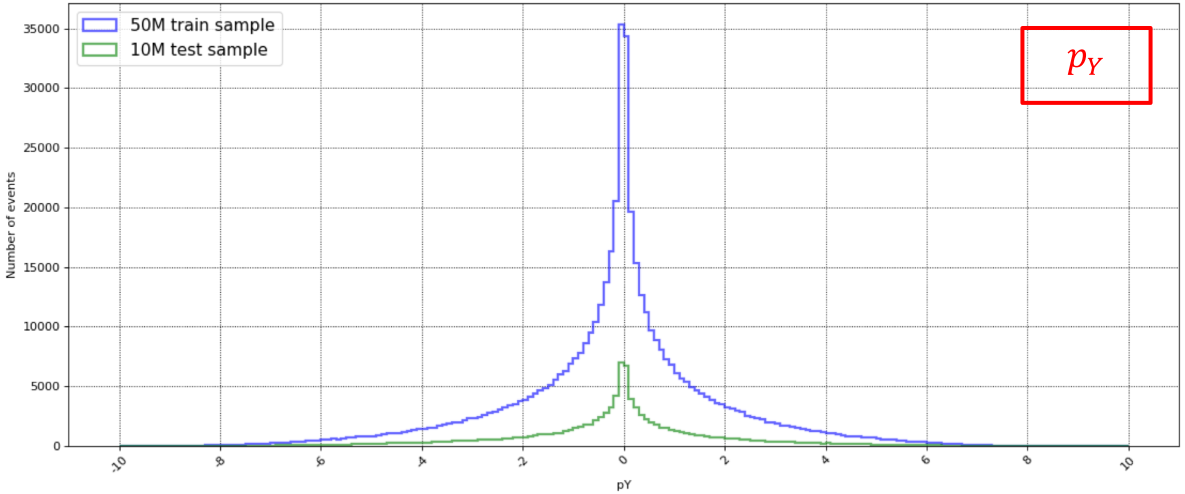
1. Input parameters changed from Energy only to [Energy, Px, Py, Pz, theta, phi]
2. Instead of checking the testing results, we compare the MC and MASK distribution in the same energy range.
3. Based on the comparison, we plot the MSE (Mean-Square-Error) loss vs Energy.
4. Optimize the MSE loss

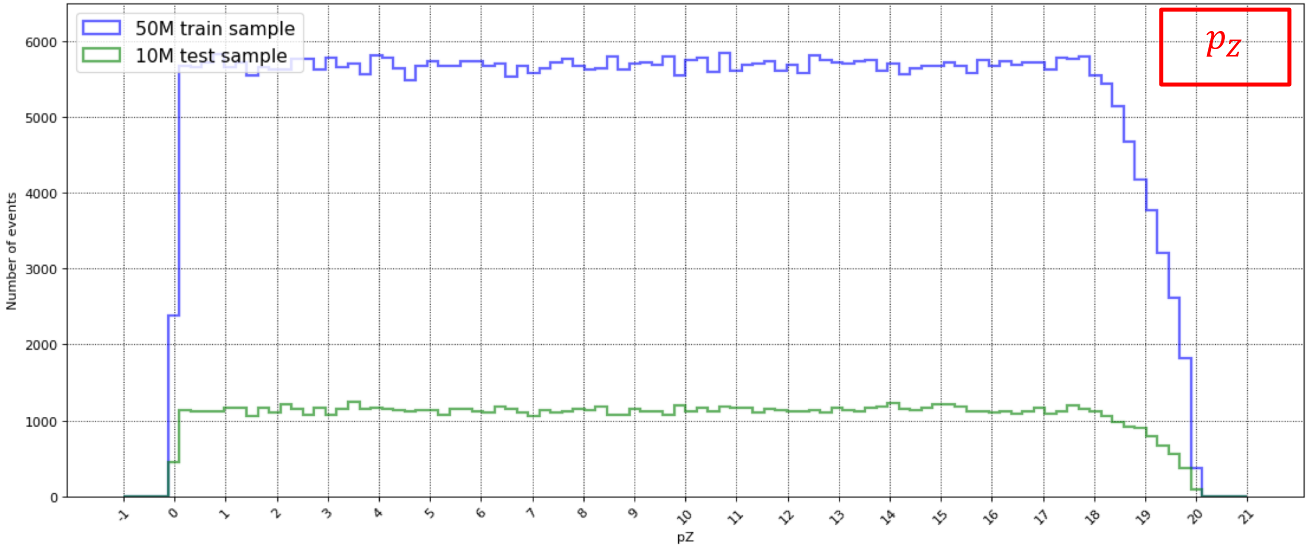
Development 1: Input distribution

We have check the input distribution to learn more about our input data.

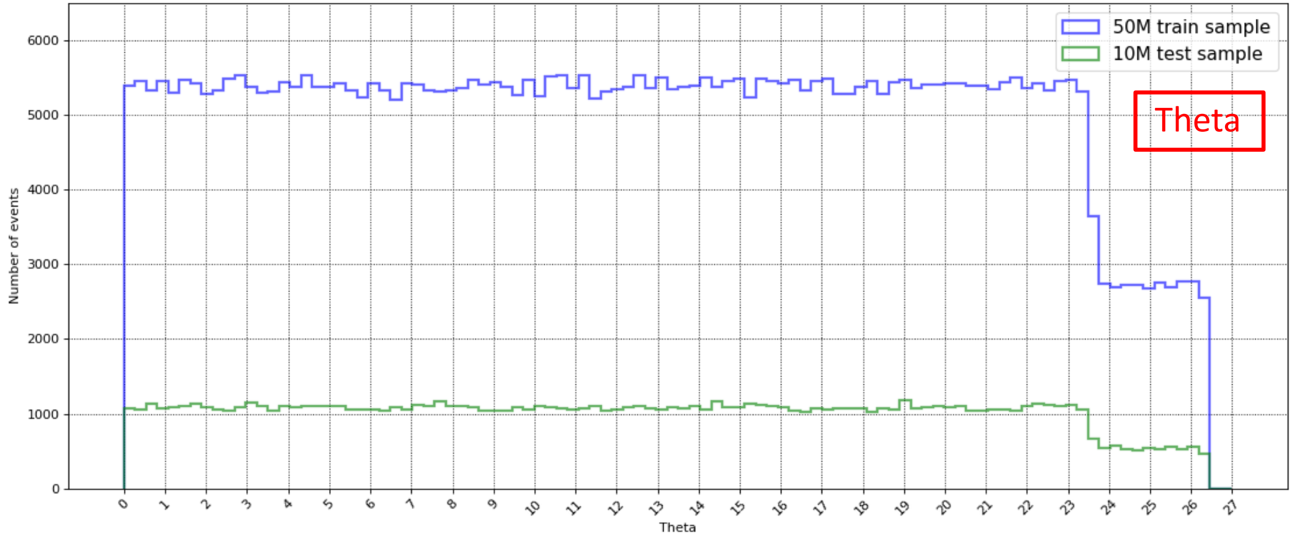


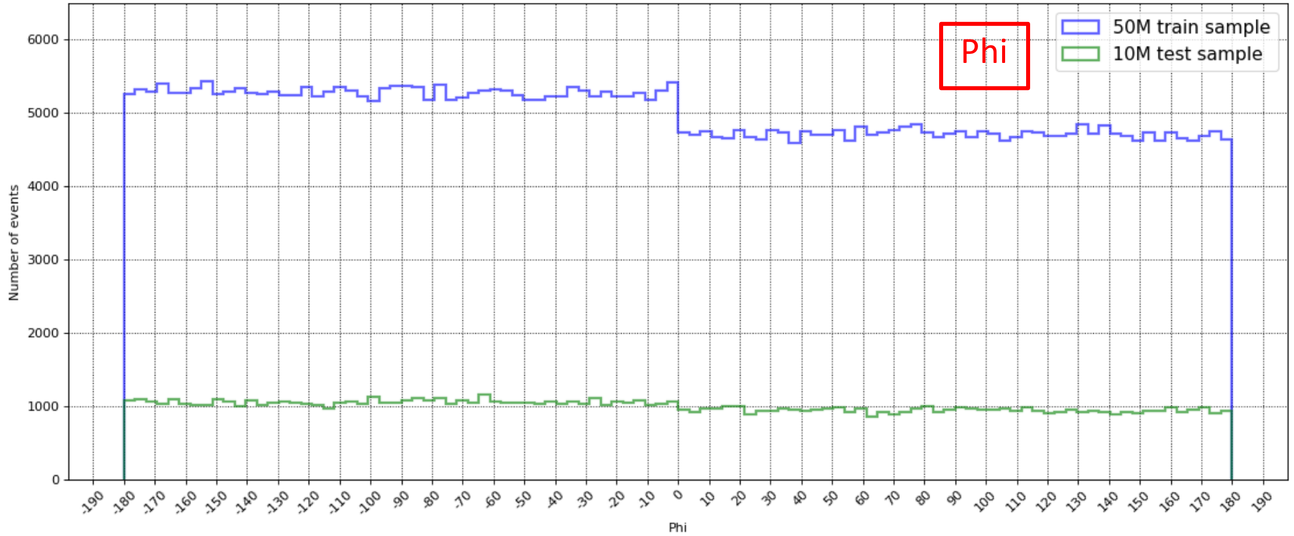






Development 1: Input distribution



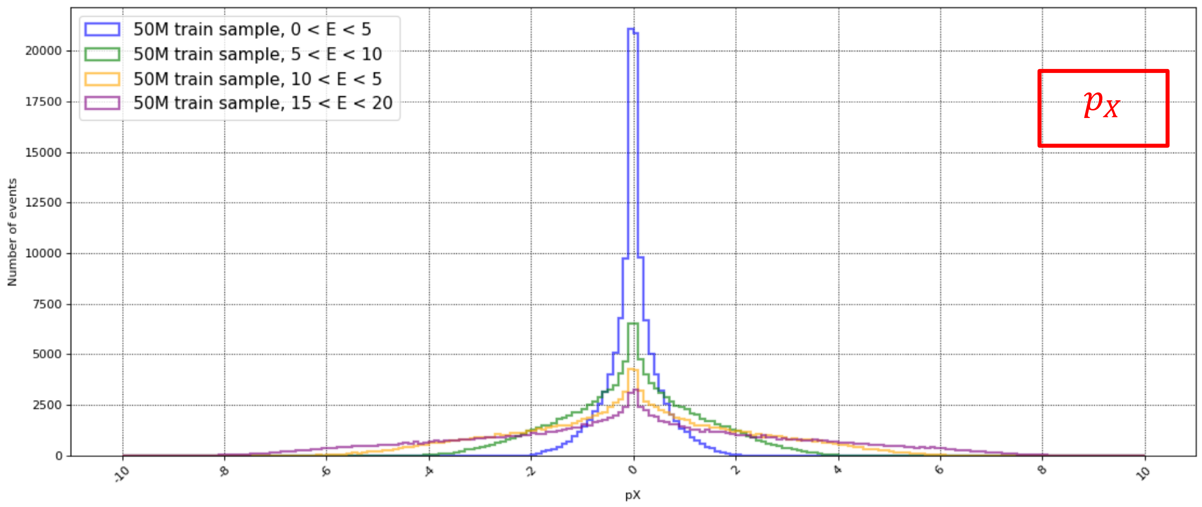


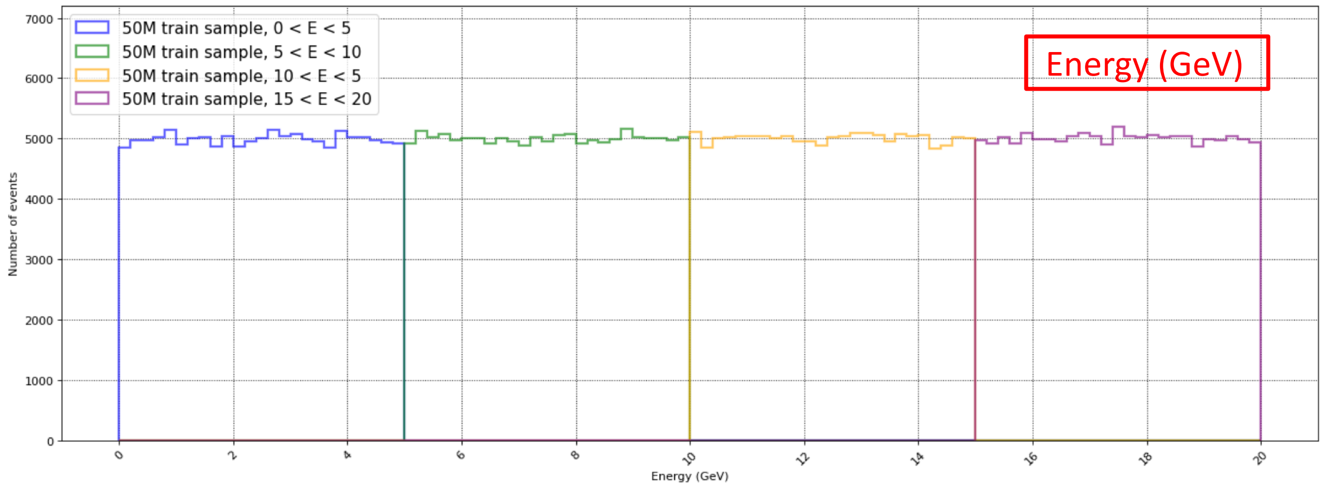

```
ionCrossingAngle = -0.025 * radian
ZDC_r_pos = 35500
ZDC_x_pos = ZDC_r_pos * math.sin(-0.025)
ZDC_y_pos = 0
ZDC_z_pos = ZDC_r_pos * math.cos(-0.025)
shift = 0
```

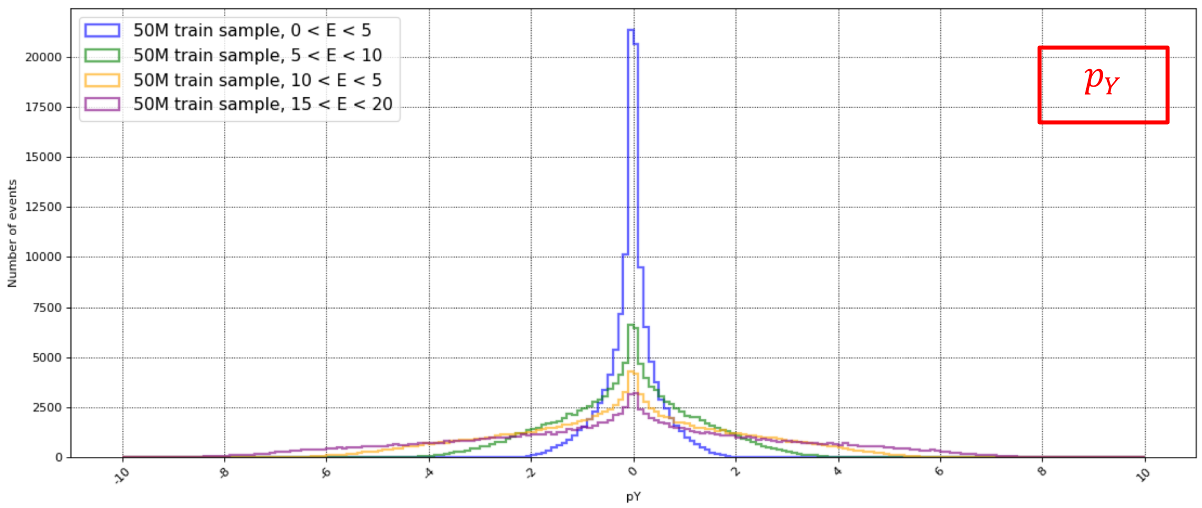
```
SIM.gun.position = (ZDC_x_pos-shift, ZDC_y_pos-shift, ZDC_z_pos)
SIM.gun.thetaMin = ionCrossingAngle - 25* degree #Later try 0
SIM.gun.thetaMax = ionCrossingAngle + 25* degree
SIM.gun.phiMin =    0* degree
#SIM.gun.phiMax =  0* degree
SIM.gun.phiMax =  180* degree
```

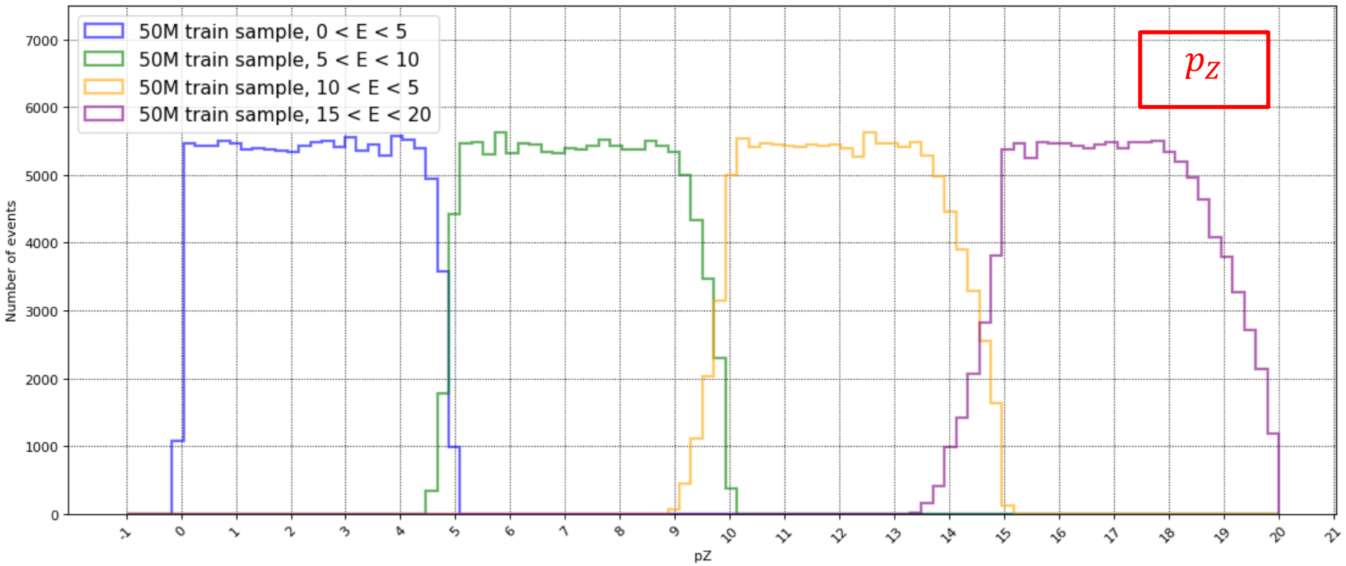
Development 1: Input distribution (Train Sample)

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

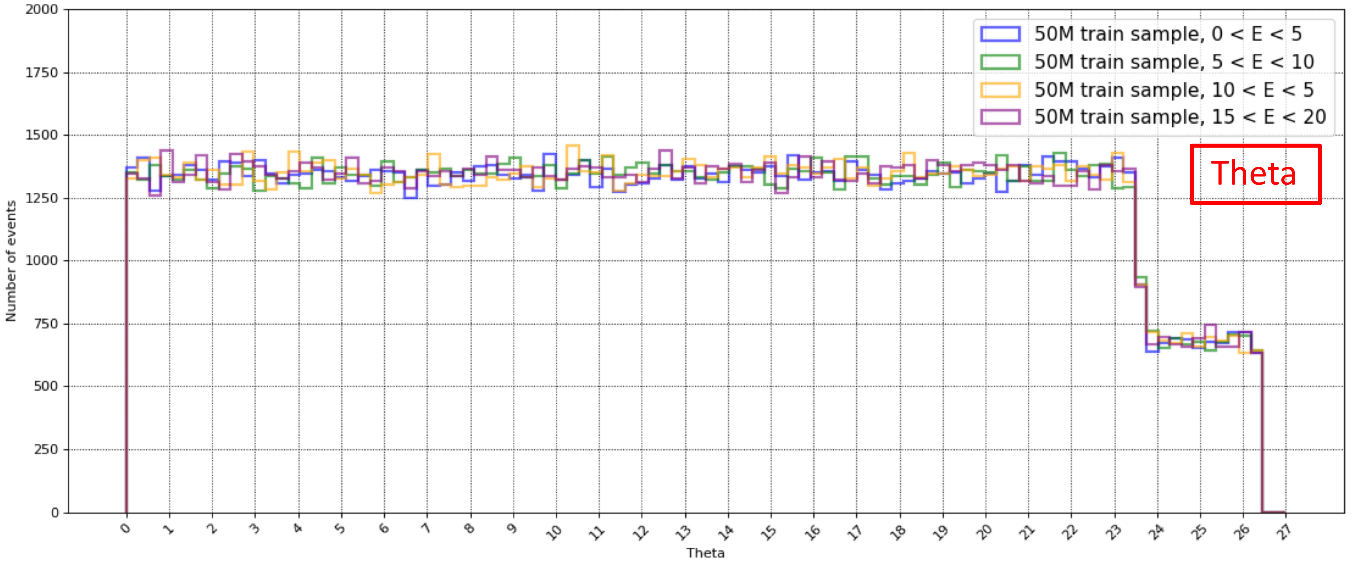


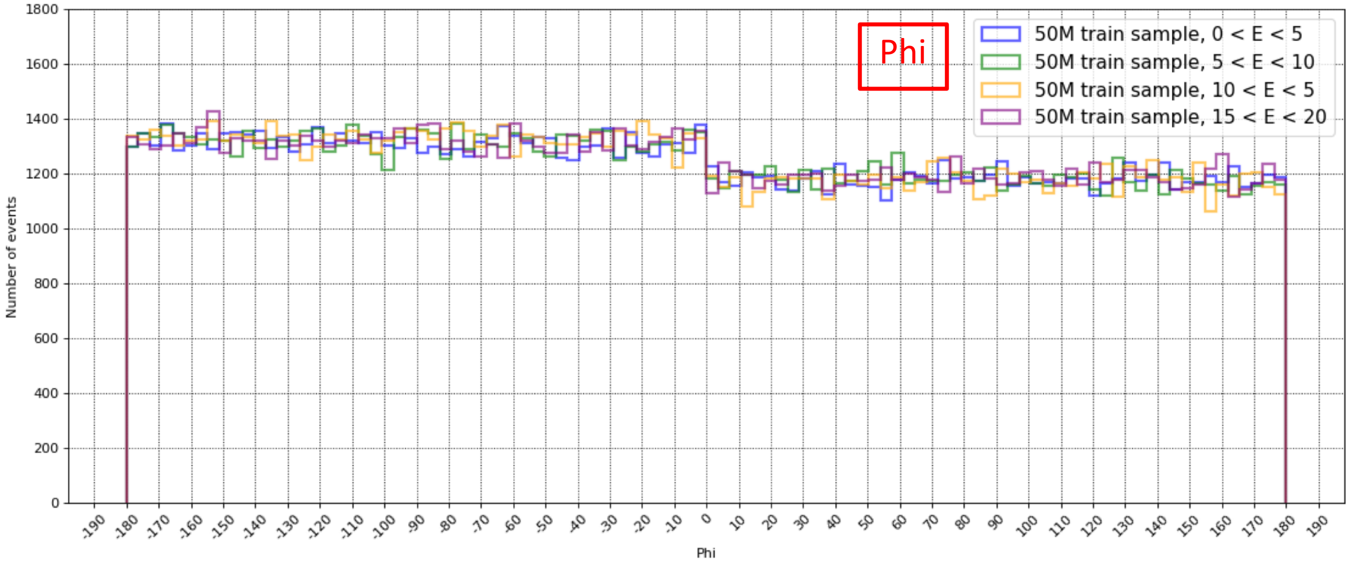






Development 1: Input distribution (Train Sample)






```
ionCrossingAngle = -0.025 * radian
ZDC_r_pos = 35500
ZDC_x_pos = ZDC_r_pos * math.sin(-0.025)
ZDC_y_pos = 0
ZDC_z_pos = ZDC_r_pos * math.cos(-0.025)
shift = 0
```

```
SIM.gun.position = (ZDC_x_pos-shift, ZDC_y_pos-shift, ZDC_z_pos)
SIM.gun.thetaMin = ionCrossingAngle - 25* degree #Later try 0
SIM.gun.thetaMax = ionCrossingAngle + 25* degree
SIM.gun.phiMin =    0* degree
#SIM.gun.phiMax =  0* degree
SIM.gun.phiMax =  180* degree
```

Development 1: Normalisation

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

Input parameters changed from Energy only to [Energy, Px, Py, Pz, theta, phi].

Related normalization function has been added into the input and the encoder in the CNN.

In principle we normalize them into [0,1], which is much easier for the model to handle during the training.

```
# Separate energy (column 0) and the rest (columns 1-5)
energy = mcPar_np[:, 0].reshape(-1, 1)  # [num_entries, 1]
others = mcPar_np[:, 1:]  # [num_entries, 5]

# Normalize energy separately with min-max (to [0, 1])
energy_min = energy.min()
energy_max = energy.max() + 1e-8  # Avoid division by zero if max == min
normalized_energy = (energy - energy_min) / (energy_max - energy_min)

# Normalize the rest with z-score
others_mean = others.mean(axis=0)
others_std = others.std(axis=0) + 1e-8
normalized_others = (others - others_mean) / others_std
```

Mathematical Formula

For each energy value E_i in E:

$$E_{\mathrm{norm},i} = \frac{E_i - E_{\min}}{E_{\max} - E_{\min}}$$

where:

- $E_{\min} = \min(E)$: the minimum energy value across all events.
- $E_{\max} = \max(E) + 10^{-8}$: the maximum energy value, with a small constant (10^{-8}) added to avoid division by zero if $E_{\max} = E_{\min}$.
- $E_{\mathrm{norm},i}$: the normalized energy value, guaranteed to be in $[0, 1]$.

Mathematical Formula

For each parameter $O_{i,j}$ in O, where i is the event index and j is the parameter index (1 to 5):

$$O_{\mathrm{norm},i,j} = \frac{O_{i,j} - \mu_j}{\sigma_j}$$

where:

- $\mu_j = \mathrm{mean}(O[:, j])$: the mean of the j-th parameter across all events.
- $\sigma_j = \mathrm{std}(O[:, j]) + 10^{-8}$: the standard deviation of the j-th parameter, with a small constant (10^{-8}) added to avoid division by zero if $\sigma_j = 0$.
- $O_{\mathrm{norm},i,j}$: the normalized value of the j-th parameter for the i-th event.

This results in each column of O_{norm} having a mean of 0 and a standard deviation of approximately 1 (unless clipped by the small constant).

Development 1 : Pre-encoding

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

We also added the Feature-wise Linear Modulation (FiLM) layer in order to shift the input features without changing the dim, and added a ReLU function to transform the encoder input (As the encoder didn't perform well while handling 6 inputs…)

```
self.param_encoder = nn.Sequential(
    nn.Linear(6, hidden_dim*2),  # Direct to hidden_dim
    nn.ReLU(),
    nn.Linear(hidden_dim*2, hidden_dim*4)  # Ensure output is hidden_dim
)

self.encoder = nn.Sequential(
        #nn.Linear(6, hidden_dim*4),
        LinearBlock(hidden_dim*4, hidden_dim*4, 4),
        LinearBlock(hidden_dim*4, hidden_dim*9, 4),
        nn.Unflatten(1, (hidden_dim, 3, 3)), # (3, 3)
        nn.ConvTranspose2d(hidden_dim, hidden_dim, kernel_size = (3, 3),
        stride = (1, 1), padding = (0, 0)), # (5, 5)
        Conv2dBlockH3W3(hidden_dim, hidden_dim*2),
        Conv2dBlockH3W3(hidden_dim*2, hidden_dim*4),
        PixelShuffle2D(2, 2), # (10, 10)
        Conv2dBlockH5W5(hidden_dim, hidden_dim*2),
        Conv2dBlockH5W5(hidden_dim*2, hidden_dim*4),
        PixelShuffle2D(2, 2), # (20, 20)
        Conv2dBlockH5W5(hidden_dim, hidden_dim),
        Conv2dBlockH5W5(hidden_dim, hidden_dim)
        )

self.film_cond_linear = nn.Linear(hidden_dim*4, hidden_dim)
self.film = FiLM(hidden_dim)
```

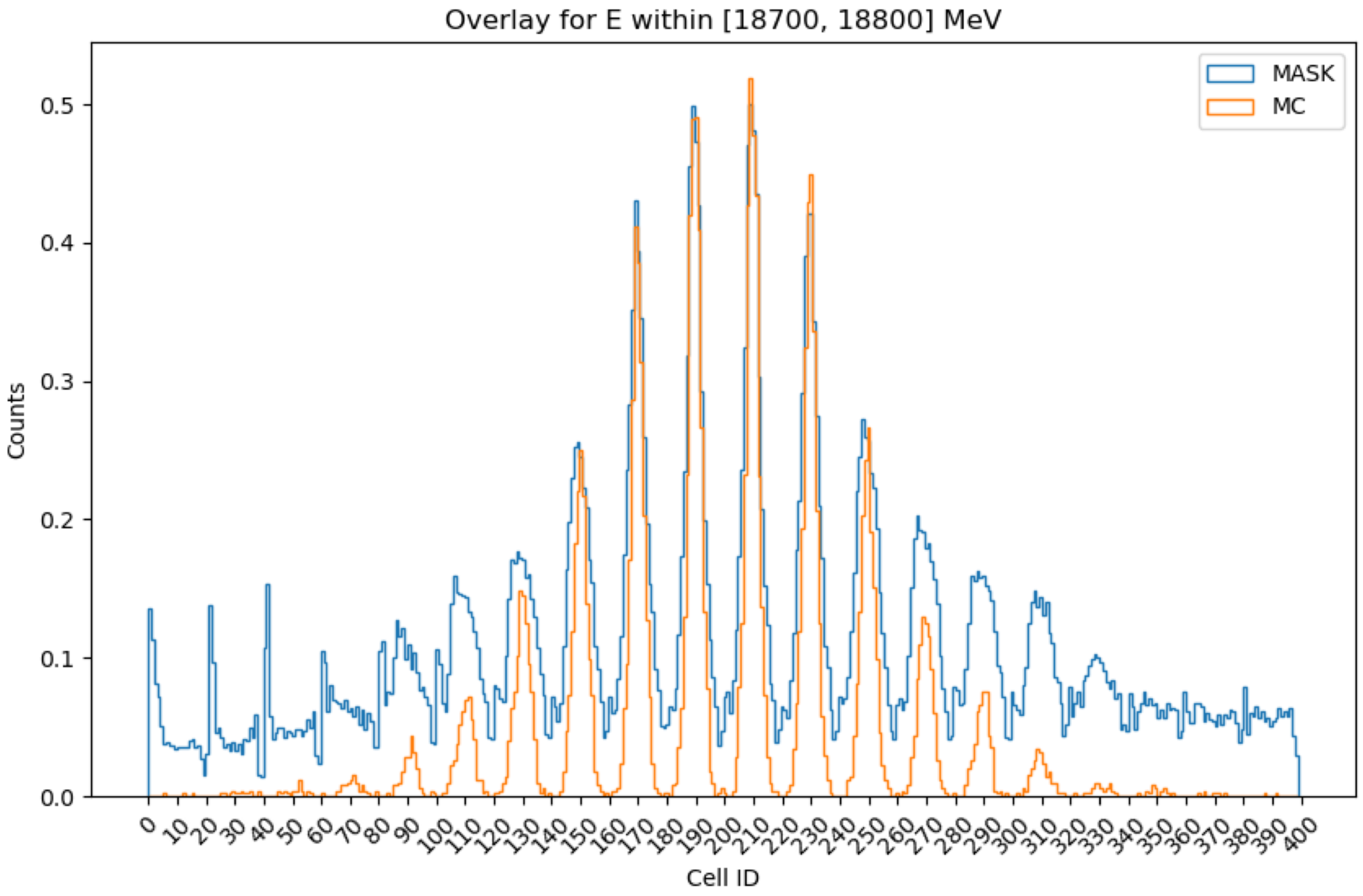
```
class FiLM(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.scale = nn.Linear(dim, dim)  # Scale from encoded params, dim=hidden_dim
        self.shift = nn.Linear(dim, dim)  # Shift from encoded params

    def forward(self, x, cond):
        # cond should be [batch_size, hidden_dim] from param_encoder
        scale = self.scale(cond).unsqueeze(-1).unsqueeze(-1)  # [batch, dim, 1, 1]
        shift = self.shift(cond).unsqueeze(-1).unsqueeze(-1)
        return x * (1 + scale) + shift
```

Development 2: Hit distribution per energy range

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

We compare the test results in this histogram instead.



MASK: Average Mask probability per cell, using the test sample with the trained model.

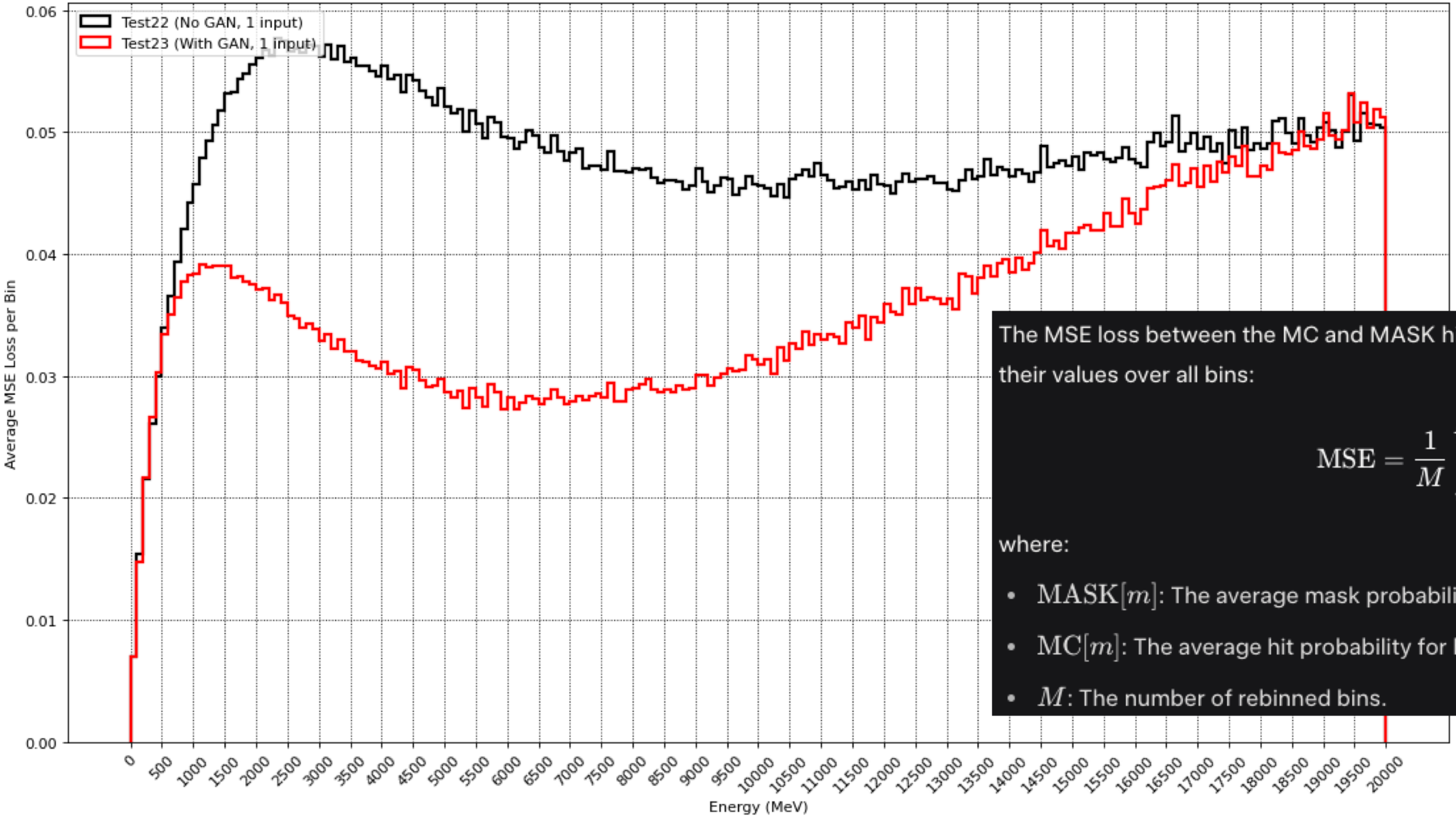
MC: Average hit probability per cell, from G4 sim directly.

Here we group events per 100 MeV.

Development 3: MSE loss definition

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

Based on the histogram we have across 0 to 20000 MeV, we can calculate the MSE loss between MC and MASK.



The MSE loss between the MC and MASK histograms is the mean of the squared differences between their values over all bins:

$$\mathrm{MSE} = \frac{1}{M}\sum_{m=0}^{M-1}\left(\mathrm{MASK}[m] - \mathrm{MC}[m]\right)^2$$

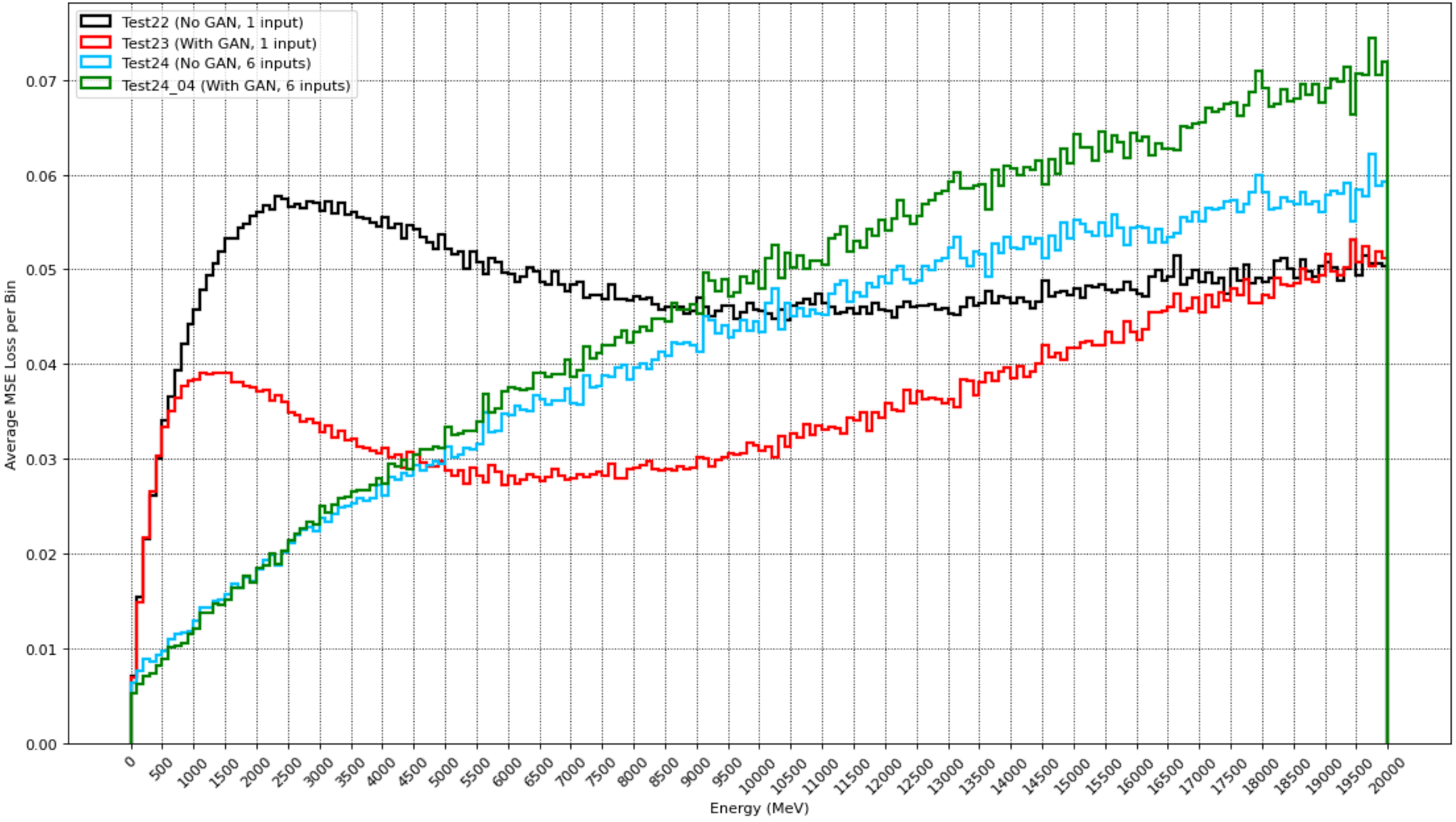
where:

- $\mathrm{MASK}[m]$: The average mask probability for bin m.
- $\mathrm{MC}[m]$: The average hit probability for bin m.
- M: The number of rebinned bins.

Development 3: MSE loss without normalisation

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

Now we can compare the MSE loss between different model.

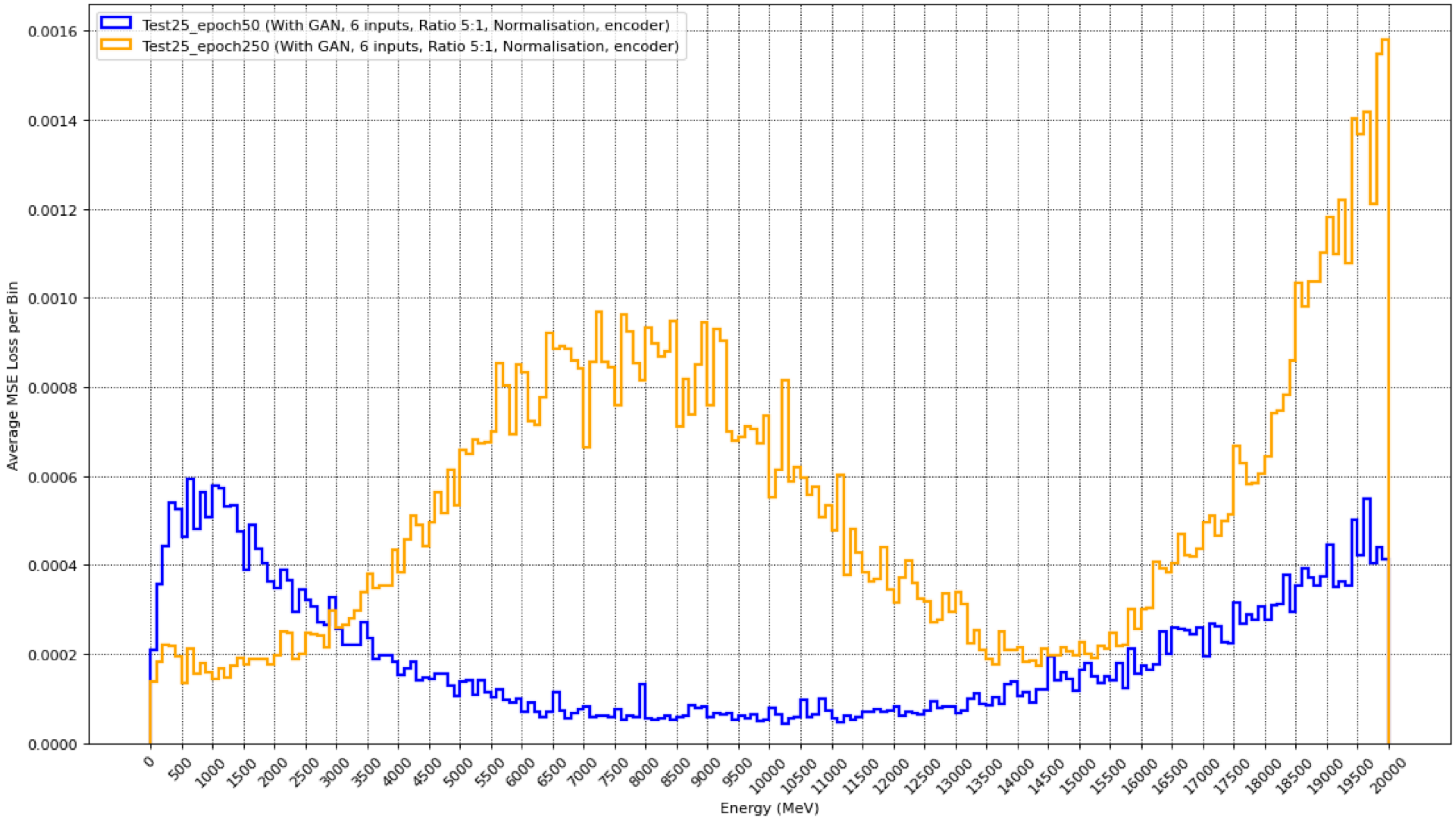


- Here Test 24 use 6 inputs but no normalization and smoothing (development 1)
- Cleanly the model didn't perform well as the energy increasing.

Development 3: MSE loss with Normalisation

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

- Applying the normalization etc, we can see the improvement (Check the scale at Y-axis)

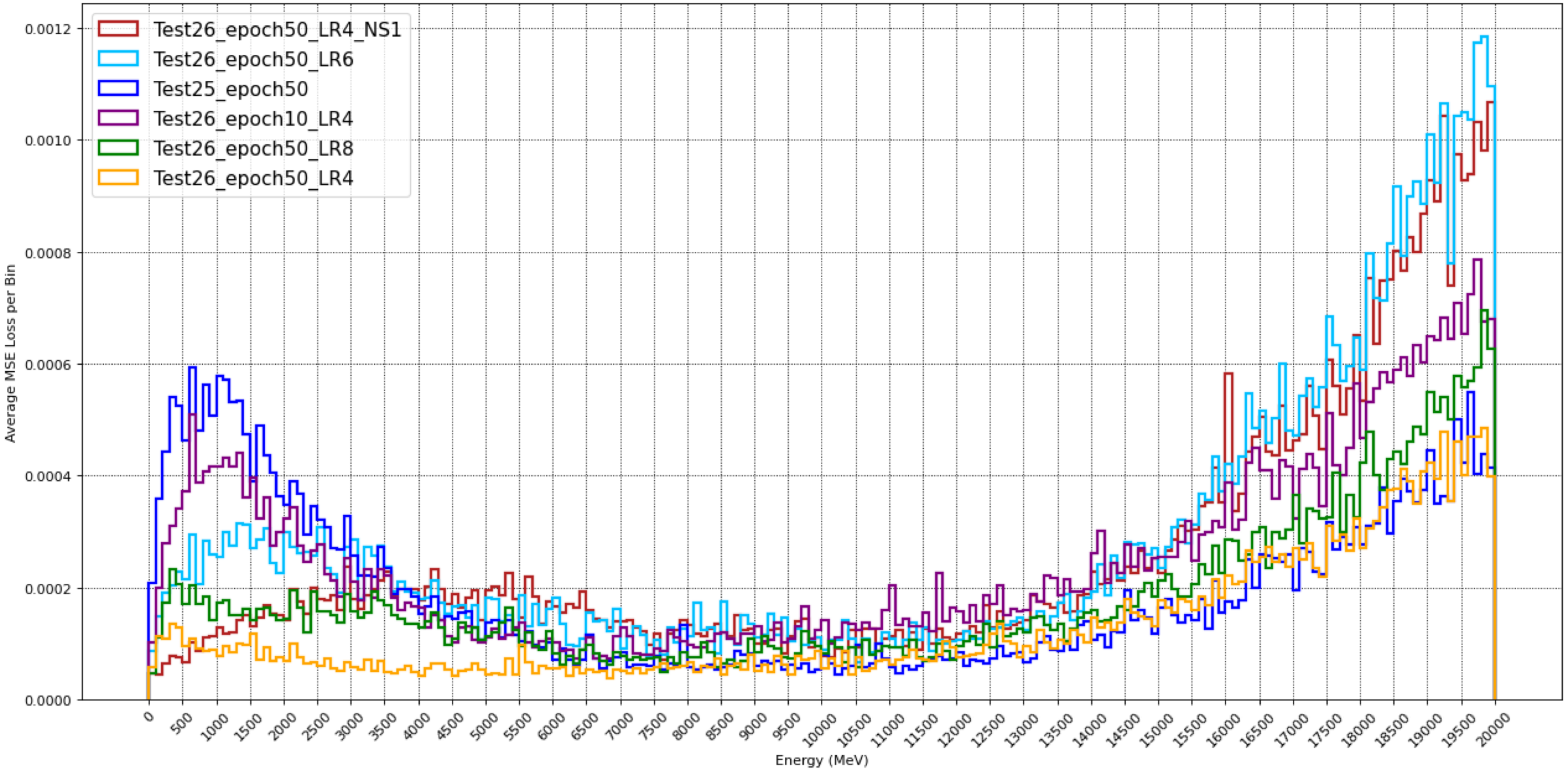


- However, with more iteration the MSE loss become worse, especially at high energy range.

Development 4: MSE loss with variable hyperparameters

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

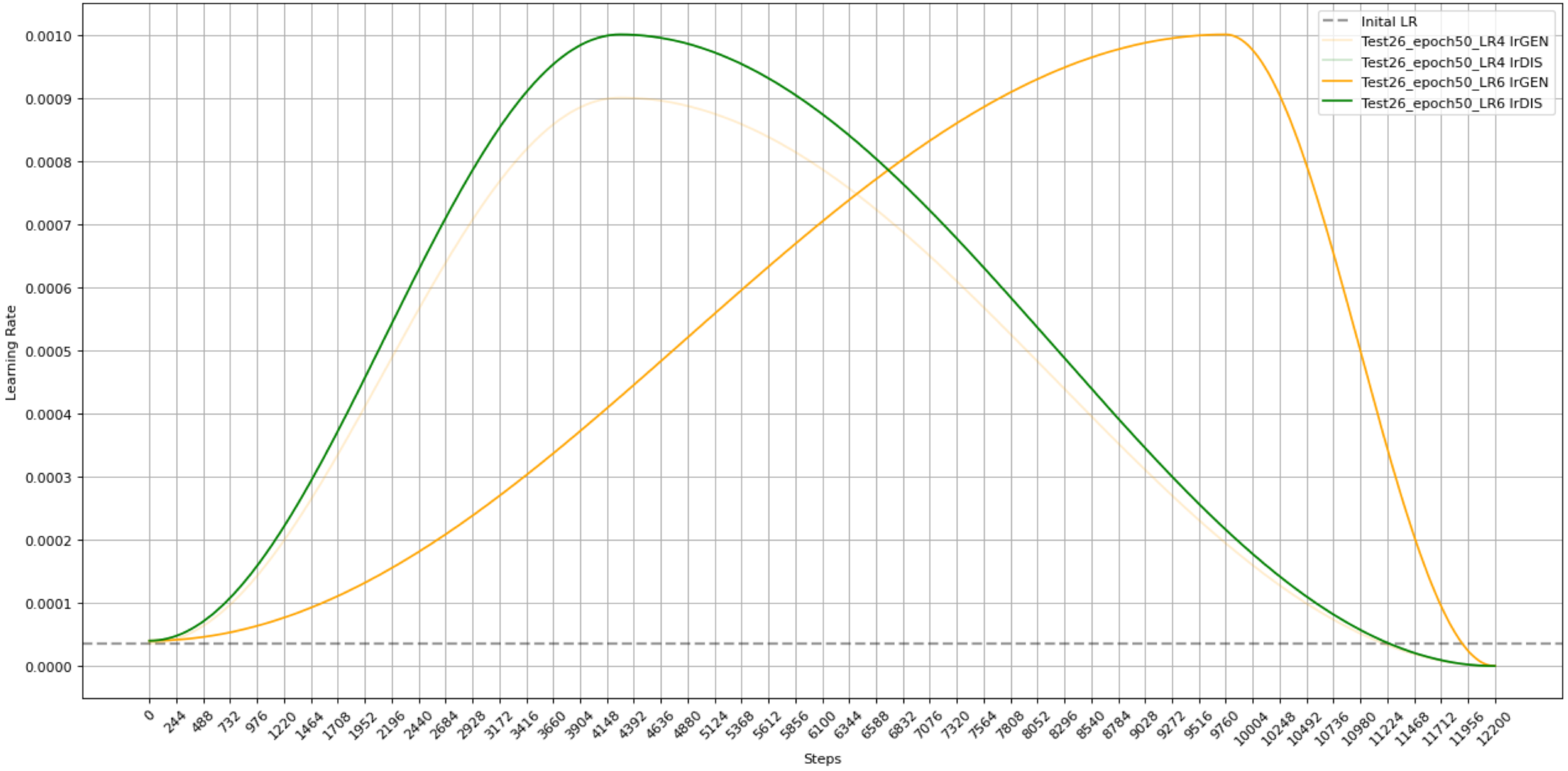
- We try to add a hyperparameter to variable the noisy level of the Generated samples along the epochs during the training, multiplying the noise strength by 0.7 to 1.0.
- Using OneCycleLR to change the LR along the iteration (See next page)



Sample Name	Hyperparameters
Test 25	Iterating DIS: GEN = 1:5
Test 26 LR4	Iterating DIS: GEN = 1:1 Max GEN LR = 9e-4 Max DIS LR = 1e-3 Pct_strat = 0.35
Test 26 LR4 NS1	Same as LR4 but noise strength always = 1.0
Test 26 LR6	Max GEN LR = 1e-3 Pct_strat = 0.8
Test 26 LR8	Max GEN LR = 7e-4

Development 4: Learning Rate cycle

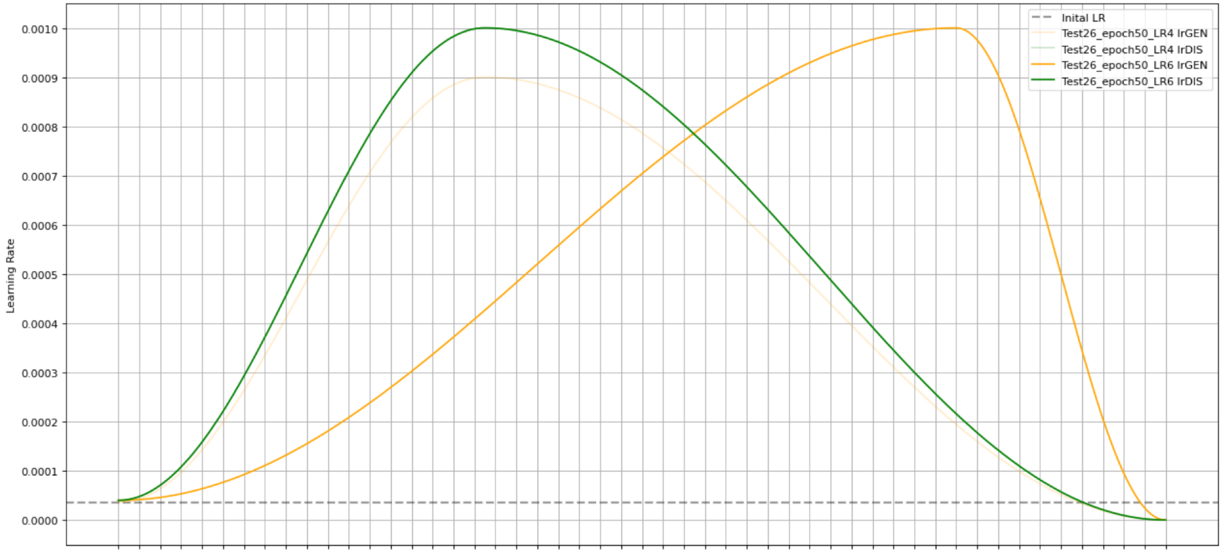
- We have 50 epochs (244 steps for 1 epoch).
- Pct_start change the timing to reach the Max LR.



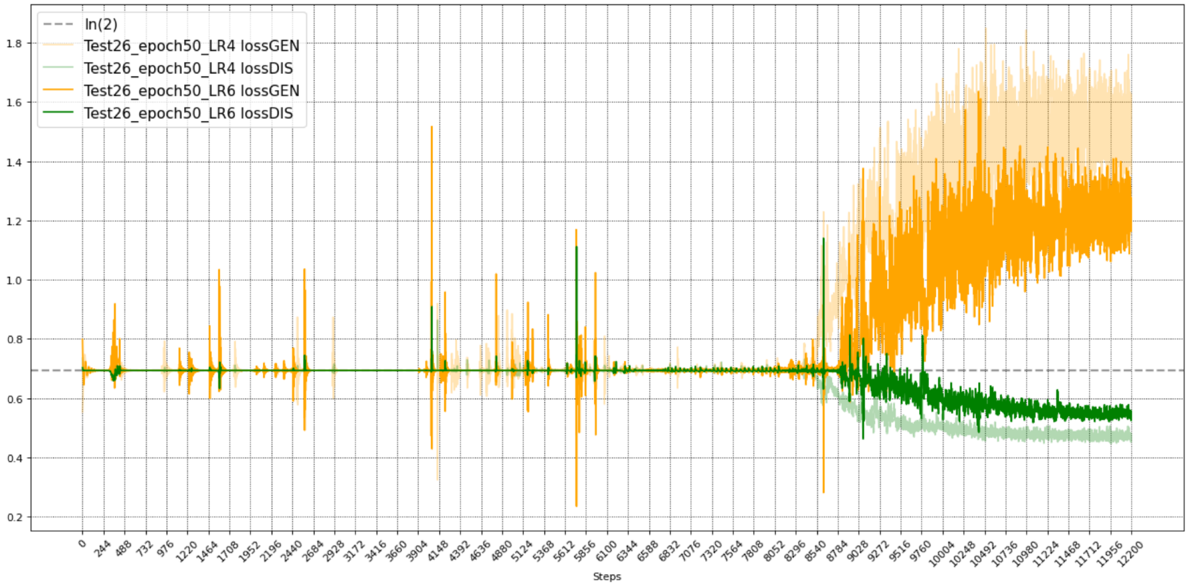
Sample Name	Hyperparameters
Test 26 LR4	Iterating DIS: GEN = 1:1 Max GEN LR = 9e-4 Max DIS LR = 1e-3 Pct_strat = 0.35
Test 26 LR6	Max GEN LR = 1e-3 Pct_strat = 0.8

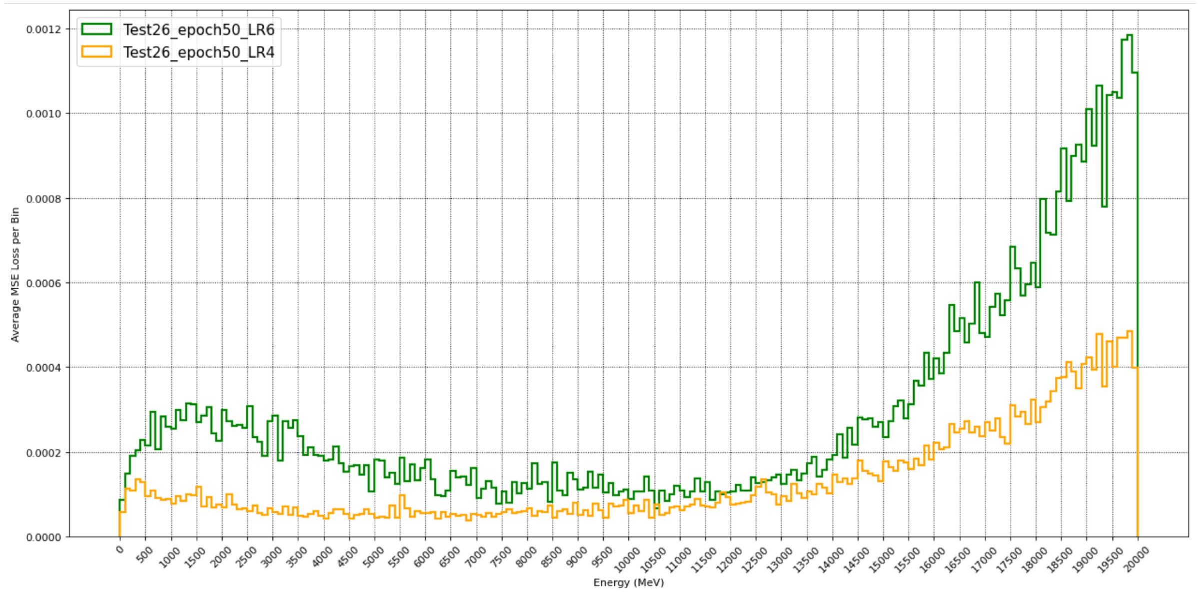
Development 4: Learning Rate vs loss GEN/DIS (LR4 v LR6)

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA



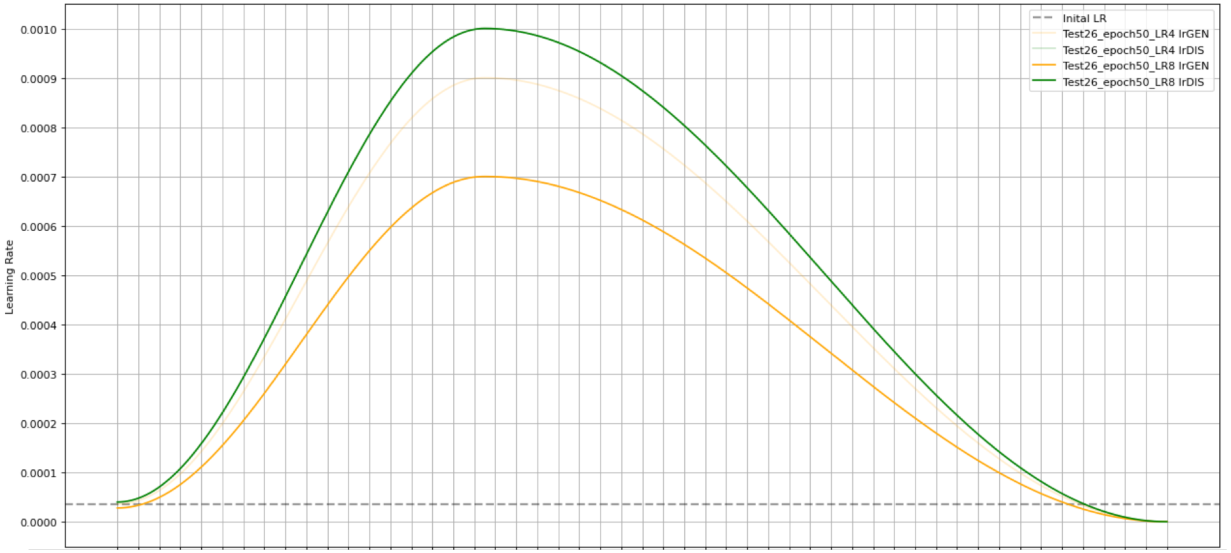
- We can see a little bit improvement by changing the LR cycle (diverges delay ~ 2 epochs)
- But the MSE loss become worse overall.

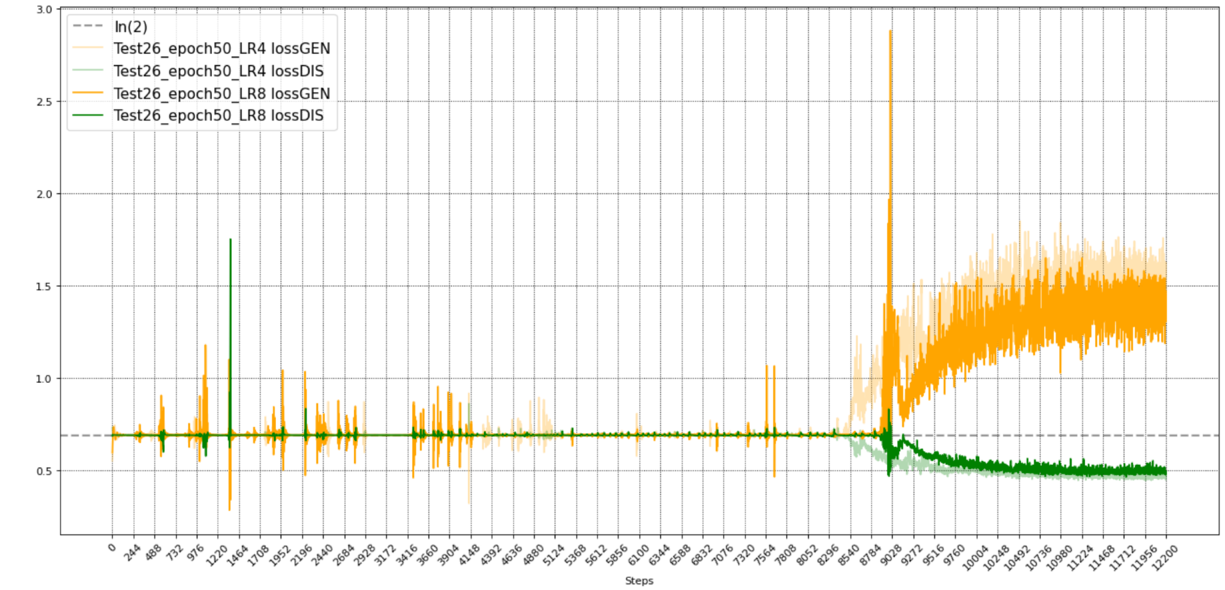




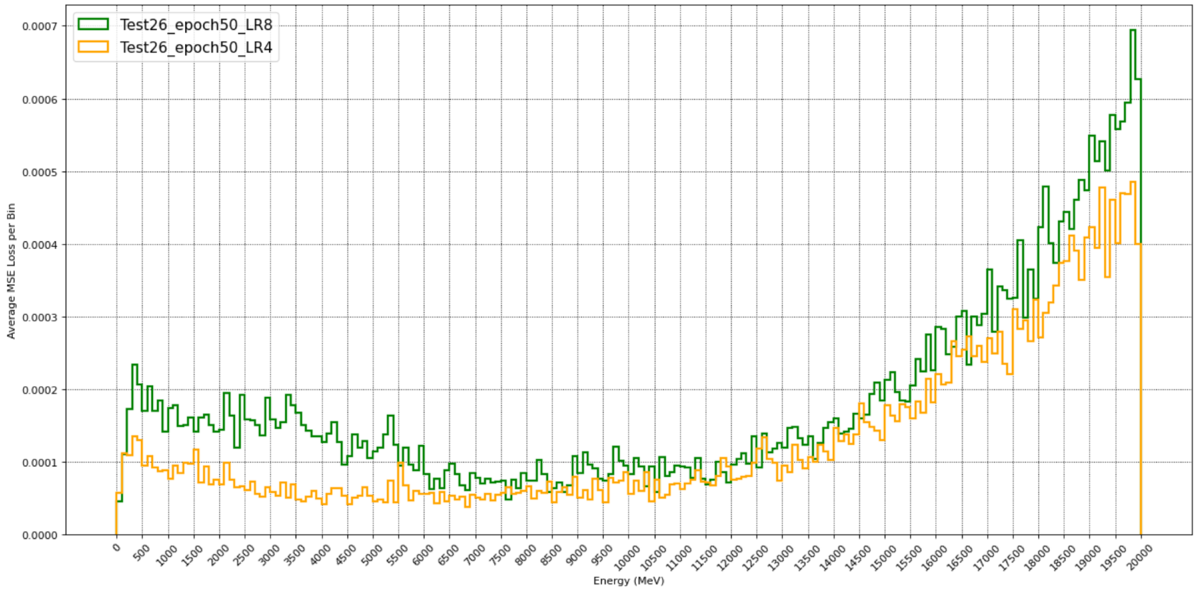
Development 4: Learning Rate vs loss GEN/DIS (LR4 v LR6)

中央研究院物理研究所 INSTITUTE OF PHYSICS, ACADEMIA SINICA





- We can see that the loss GEN/DIS was trying to converge but eventually failed then diverge.
- The MSE loss got worse at the low and high energy range.

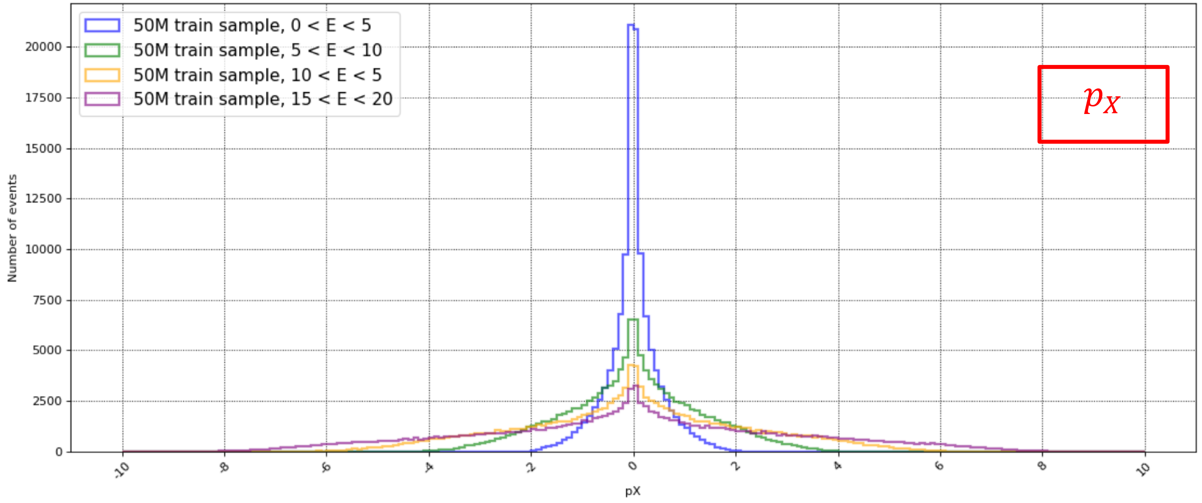


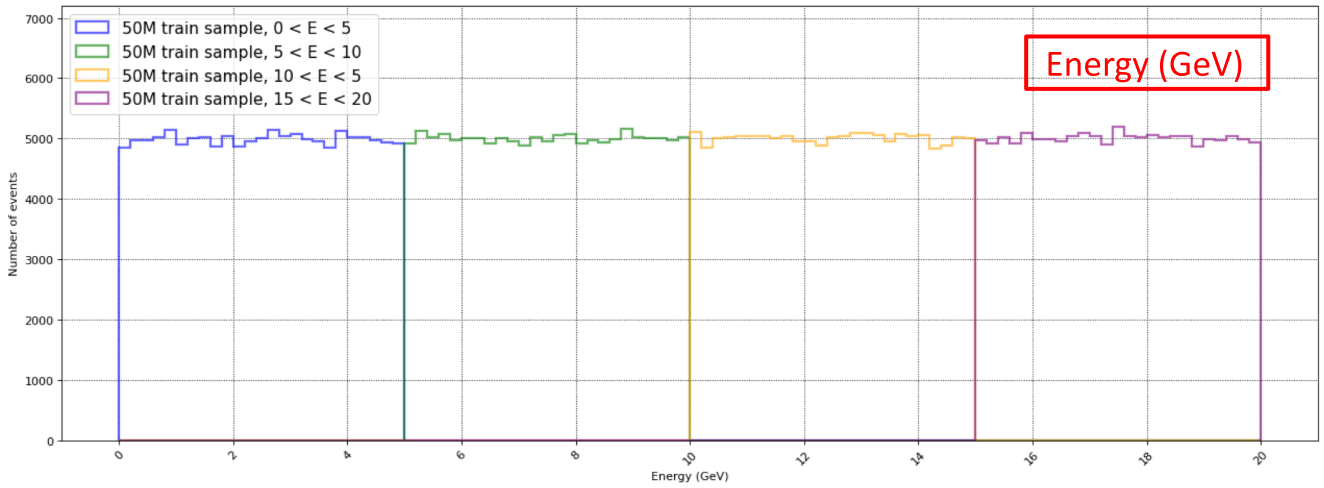
Outlook

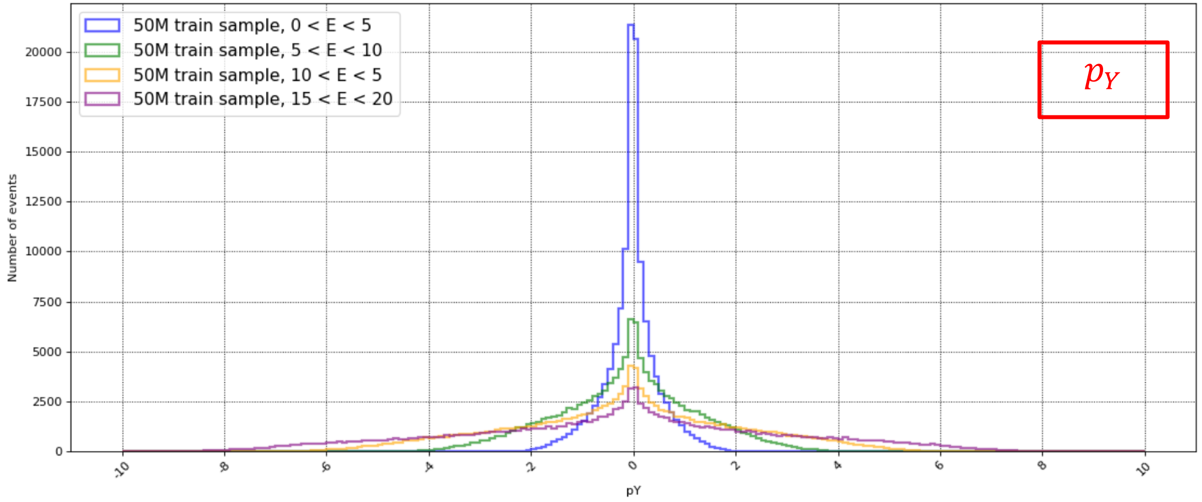
- We can now use the MSE loss to check the energy dependence in the model.
- We try to optimize the MSE loss, as the MSE loss become worse at high energy range always.
- In fact the MSE loss alright low enough (< 0.002 in general), and we can't easily improve the MSE loss at the high energy range.
- On-going task:
  - Try to reverse the Max LR for GEN and DIS (LR9)
  - Generating a much larger samples (250M)
  - Remove the GAN part and see the training result with regression part only.
- If LR9 didn't work then we are gonna use LR4 setting and see can we improve the MSE loss at high energy range with training with 250M samples.

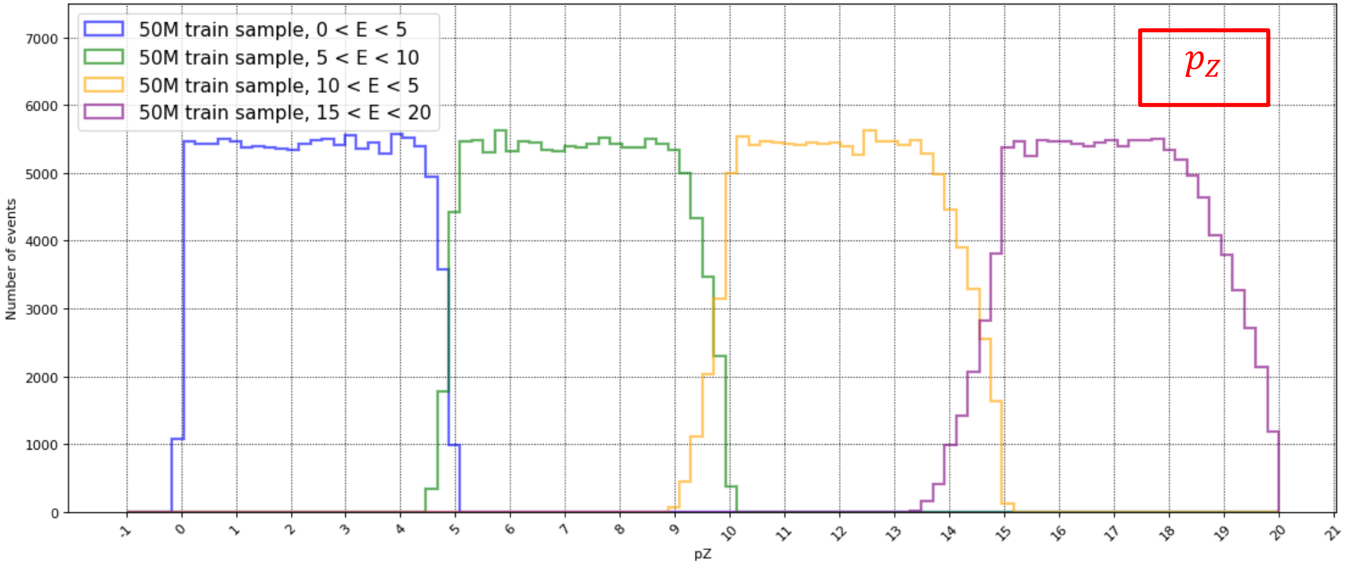
Development 1 (Test Sample)

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

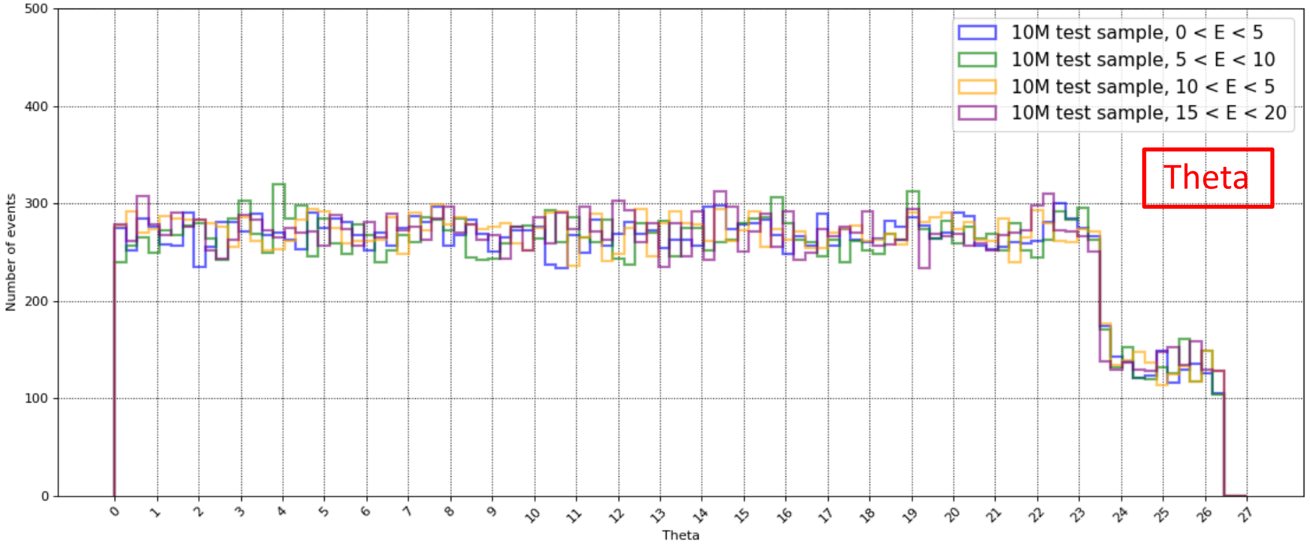


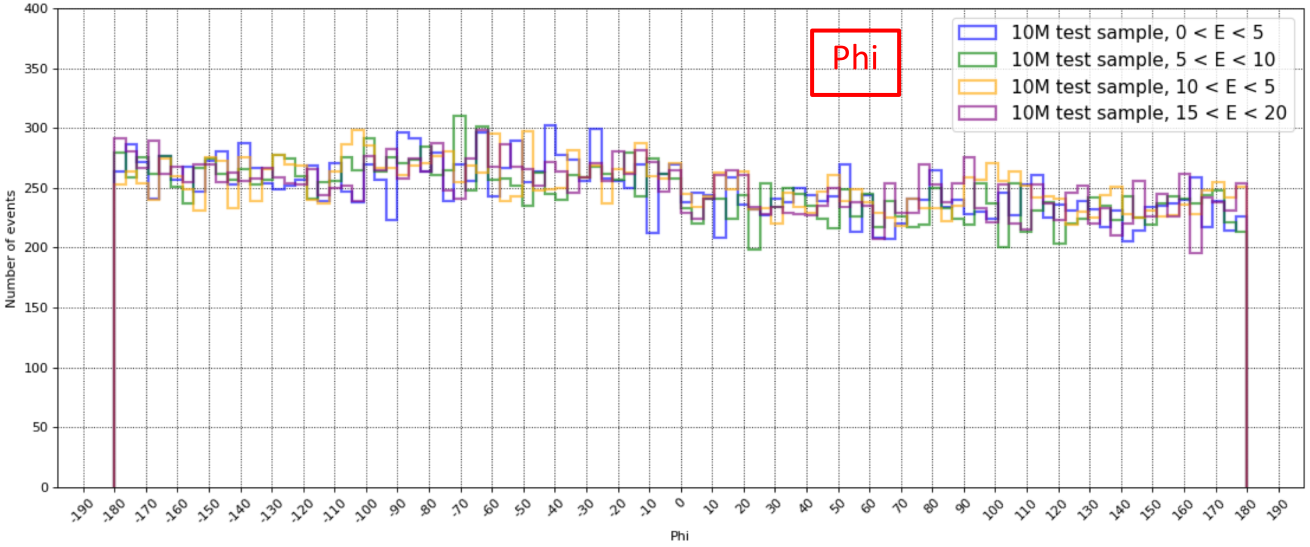






Development 1 (Test Sample)



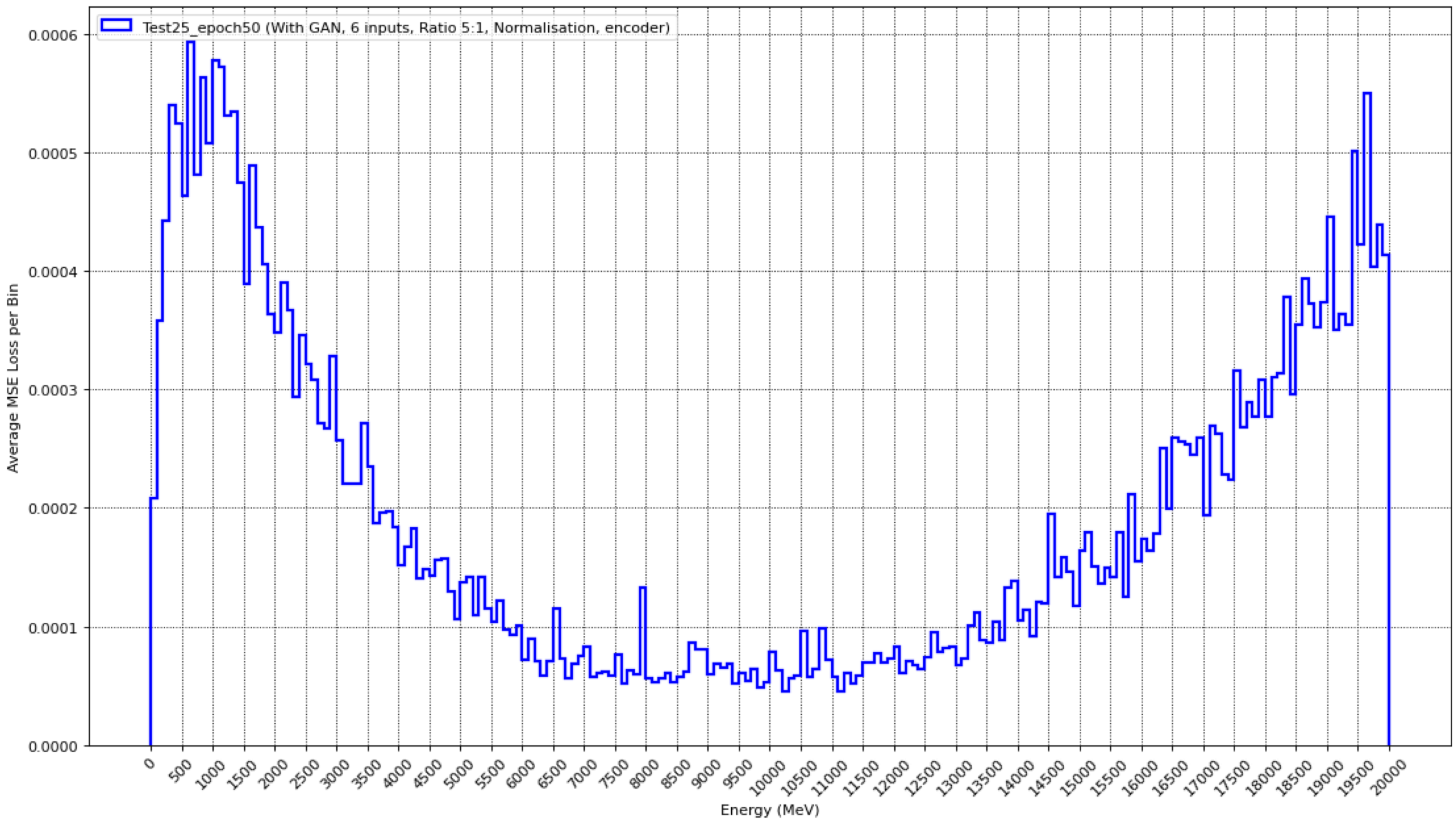



```
ionCrossingAngle = -0.025 * radian
ZDC_r_pos = 35500
ZDC_x_pos = ZDC_r_pos * math.sin(-0.025)
ZDC_y_pos = 0
ZDC_z_pos = ZDC_r_pos * math.cos(-0.025)
shift = 0
```

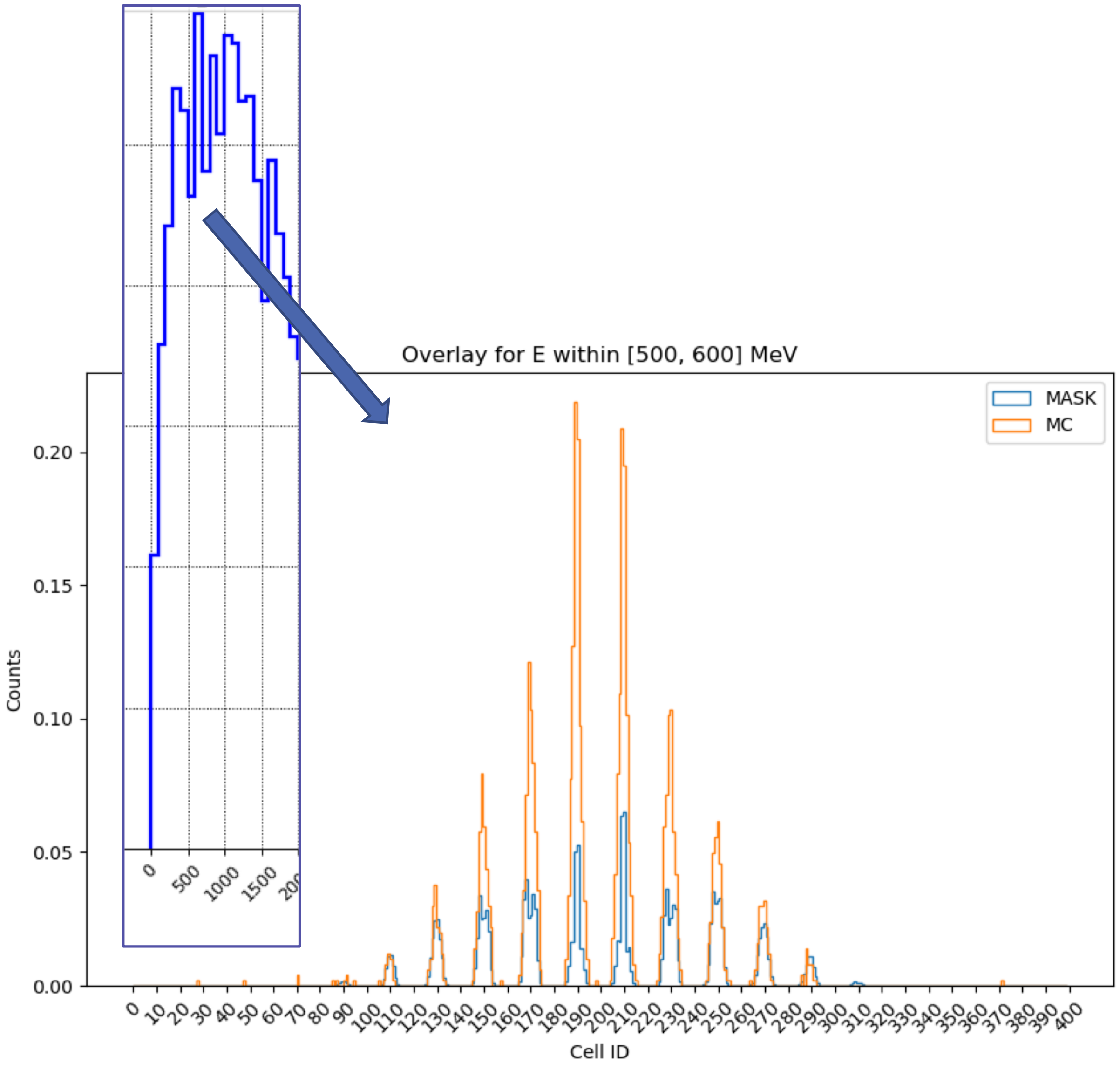
```
SIM.gun.position = (ZDC_x_pos-shift, ZDC_y_pos-shift, ZDC_z_pos)
SIM.gun.thetaMin = ionCrossingAngle - 25* degree #Later try 0
SIM.gun.thetaMax = ionCrossingAngle + 25* degree
SIM.gun.phiMin =    0* degree
#SIM.gun.phiMax =   0* degree
SIM.gun.phiMax =  180* degree
```

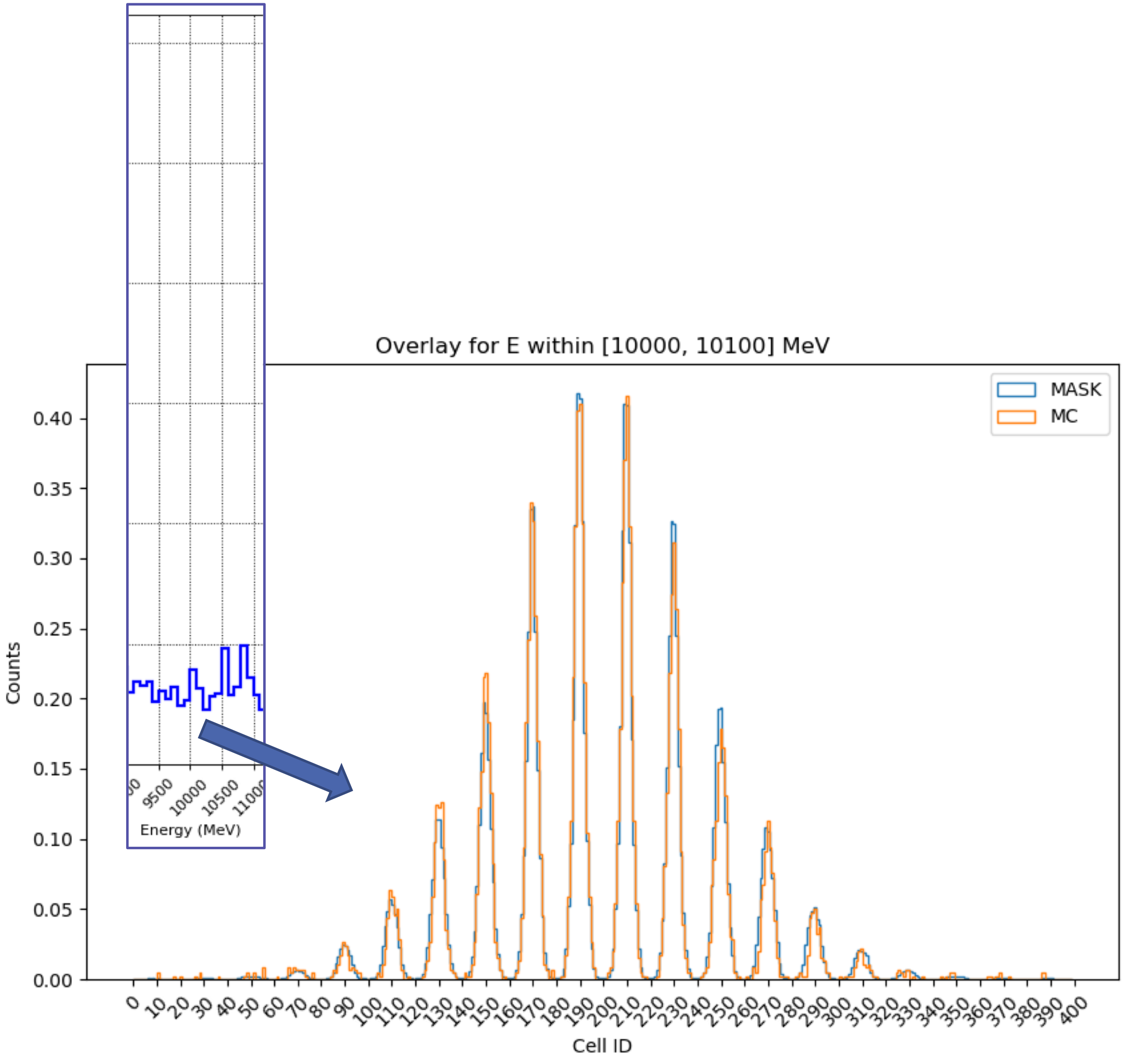
Development 3

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA



Development 3

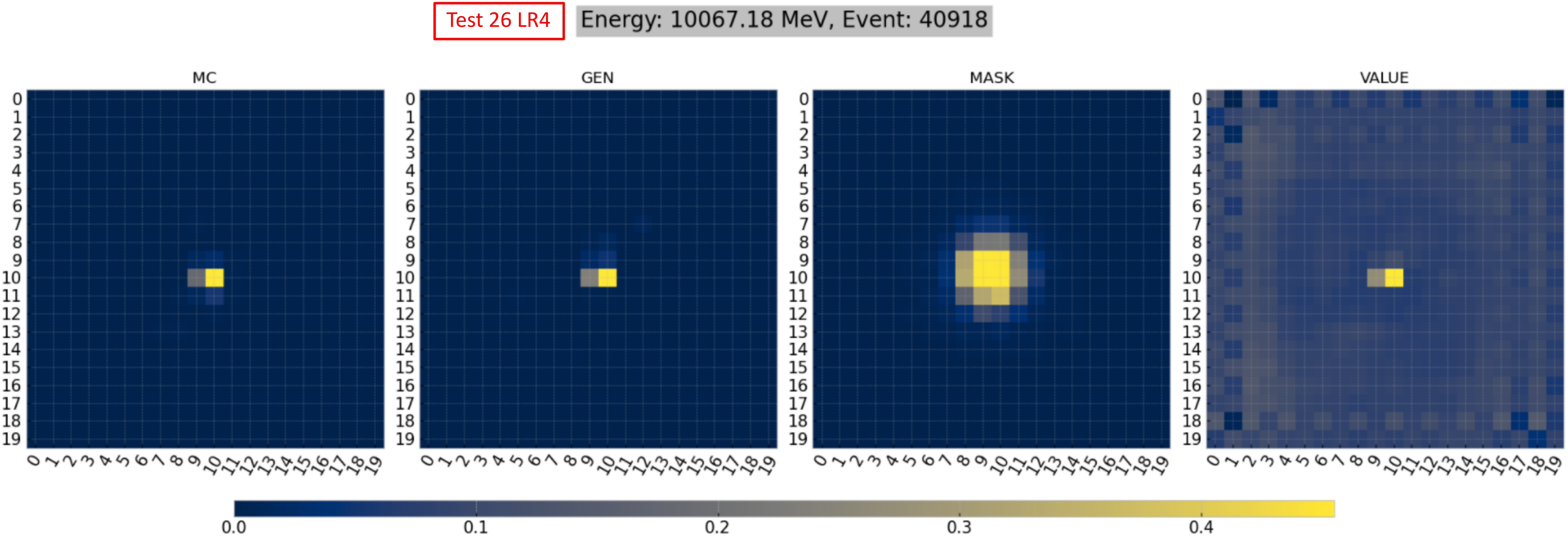




Development 3: Test result (no GAN)

中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

Test 26 LR4



Development 3: Test result (with GAN)