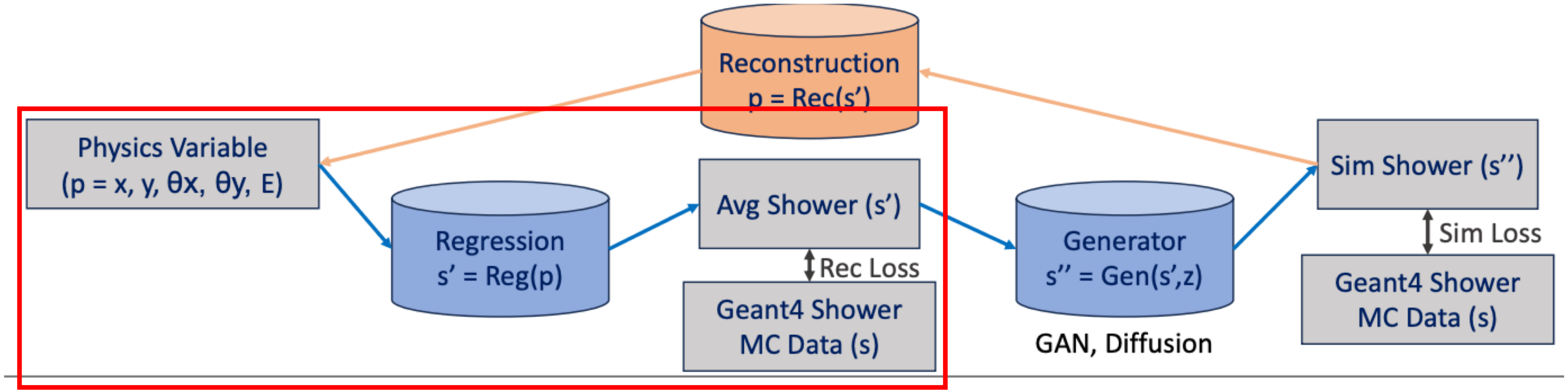
中央研究院物理研究所
INSTITUTE OF PHYSICS, ACADEMIA SINICA

# Status report

2025/06/19
ZDC Internal
WAI YUEN CHAN

# Introduction




Pre-processing (Data) → Input to → Model → Train by → Objective Function (Loss)

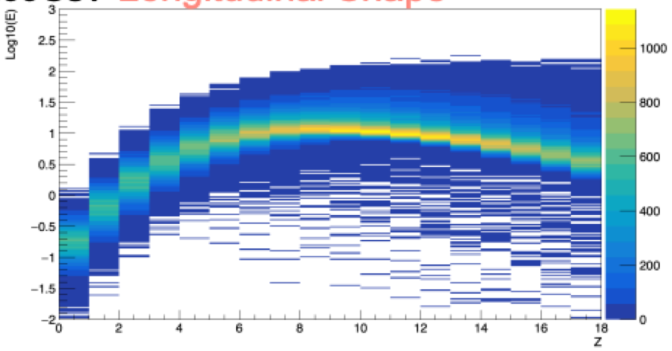
Model → Test by → Evaluation Methodology (Metric)

- Balanced samples (uniform)
- Data normalization ($\mu = 0, \sigma = 1$)

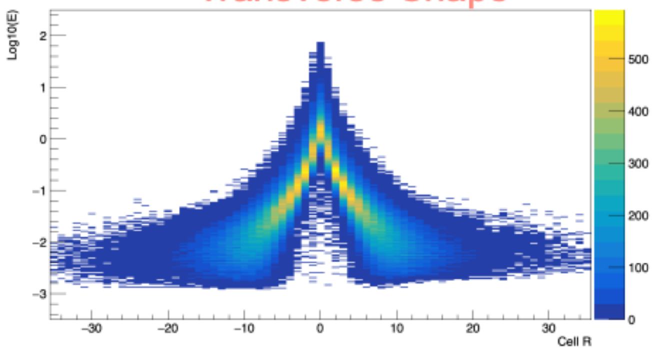
cell energy distribution

- GAN: Cross entropy loss
- REC: L2 loss

e-, 100GeV Longitudinal Shape



Transverse Shape



The longitudinal shower profile is described by the Gamma distribution:

$$\frac{dE}{dt}(t) = E_0 \frac{(\beta t)^{\beta T_0} \beta e^{-\beta t}}{\Gamma(\beta T_0 + 1)}, \quad (1)$$

using parameters described above. The scale parameter $\beta$ is found to be constant over the wide energy range, therefore it is fixed in the present analysis. Individual shower parameters $E_0$ and $T_0$ are obtained from a fit to observed energy depositions in the ECAL cells of that shower. Fig. 7 shows the high quality of the description of electron showers over a wide energy range by the model based on Eq. (1).

At a given shower depth, the transverse shower shape as a function of the distance from the shower axis $r$ is described by the sum of a narrow core and a wide tail:

$$\frac{dE}{dr}(t, r) \propto Q_C \frac{2rR_C^2}{(r^2 + R_C^2)^2} + (1 - Q_C) \frac{2rR_T^2}{(r^2 + R_T^2)^2}, \quad (2)$$

# Network structure



Hidden dim = 32 (i.e. 32 neurons in each hidden layers)
Add layers into the default structure: "Extended 2"

Default

```
self.encoder = nn.Sequential(
        nn.Linear(1, 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)
        )
```

Extended 2

```
self.encoder = nn.Sequential(
        nn.Linear(1, hidden_dim*4), #0
        LinearBlock(hidden_dim*4, hidden_dim*4, 4), #1
        LinearBlock(hidden_dim*4, hidden_dim*4, 4), #2
        LinearBlock(hidden_dim*4, hidden_dim*4, 4), #3
        LinearBlock(hidden_dim*4, hidden_dim*9, 4), #4
        nn.Dropout(0.1), #5
        nn.Unflatten(1, (hidden_dim, 3, 3)), # (3, 3) #6
        nn.ConvTranspose2d(hidden_dim, hidden_dim, k
        ernel_size = (3, 3), stride = (1, 1),
        padding = (0, 0)), # (5, 5) #7
        Conv2dBlockH3W3(hidden_dim, hidden_dim*2), #8
        Conv2dBlockH3W3(hidden_dim*2, hidden_dim*2), #9
        Conv2dBlockH3W3(hidden_dim*2, hidden_dim*2), #10
        Conv2dBlockH3W3(hidden_dim*2, hidden_dim*4), #11
        nn.Dropout(0.1), #12
        PixelShuffle2D(2, 2), # (10, 10) #13
        Conv2dBlockH5W5(hidden_dim, hidden_dim*2), #14
        Conv2dBlockH5W5(hidden_dim*2, hidden_dim*2), #15
        Conv2dBlockH5W5(hidden_dim*2, hidden_dim*2), #16
        Conv2dBlockH5W5(hidden_dim*2, hidden_dim*4), #17
        nn.Dropout(0.1), #18
        PixelShuffle2D(2, 2), # (20, 20) #19
        Conv2dBlockH5W5(hidden_dim, hidden_dim), #20
        Conv2dBlockH5W5(hidden_dim, hidden_dim), #21
        Conv2dBlockH5W5(hidden_dim, hidden_dim), #22
        Conv2dBlockH5W5(hidden_dim, hidden_dim) #23
        )
```

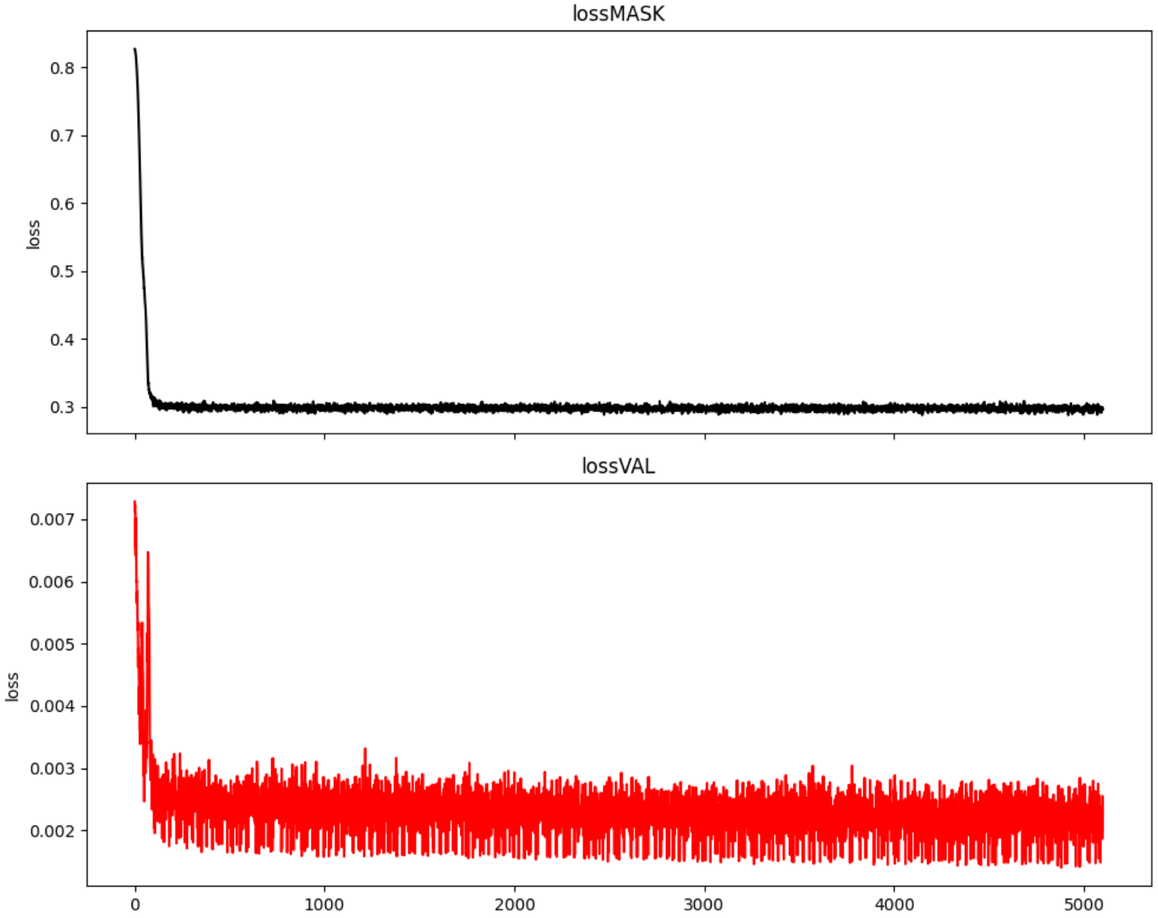
# Definition of loss

Originally we add up loss_mask, loss_val and loss_gen; also calculate the loss_dis separately.
Now we removed the GAN part so we only use the sum of loss_mask and loss_val to train the regression model.

```
#lossG = loss_mask + loss_val + loss_gen #17_02
lossG = loss_mask + loss_val
#lossG = 0.05 * loss_mask + 0.2 * loss_val + loss_gen
```

Also here we didn't add any weighting to keep the baseline clear for the upcoming test.

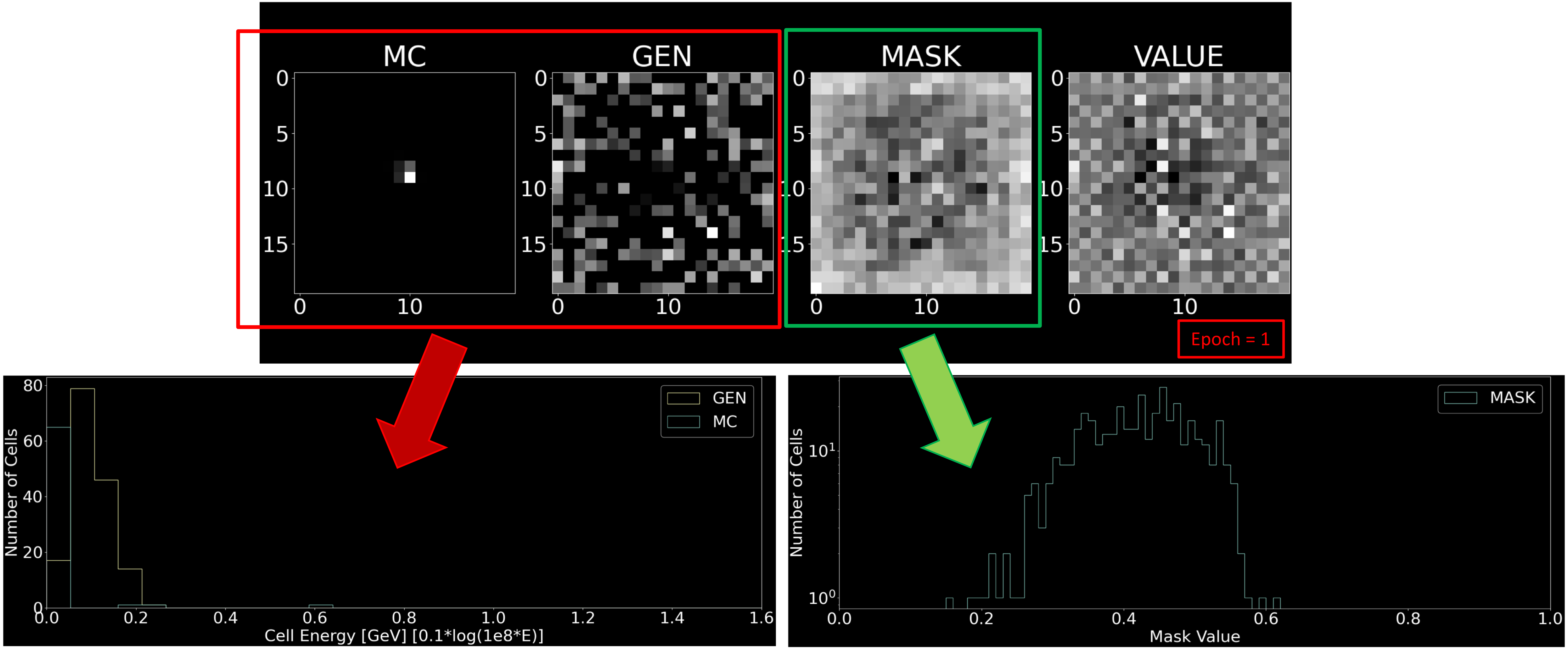
The loss function become stable at the early state.
However, the changes in the image itself is interesting.



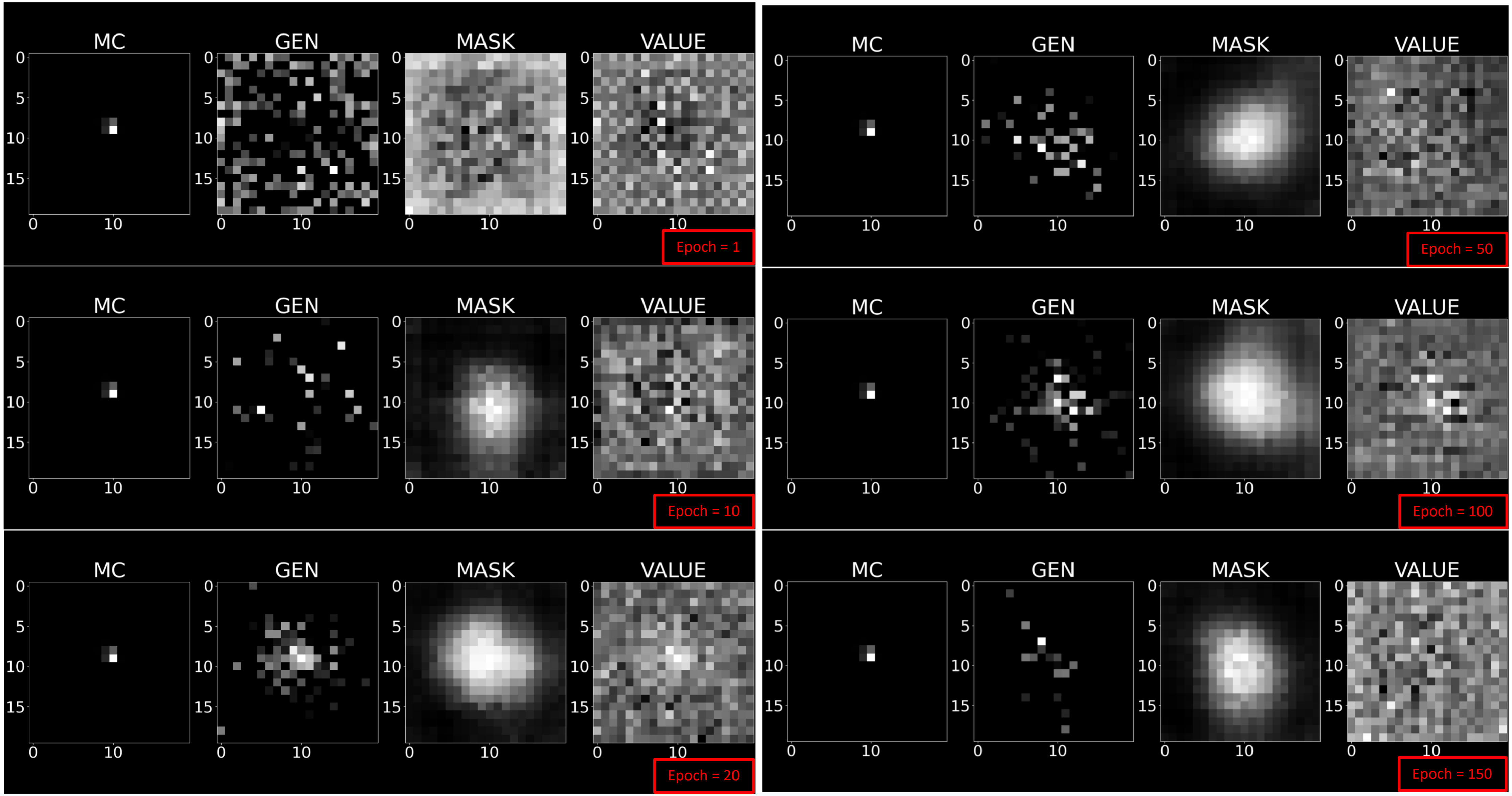
# Test result (image)

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

Besides the image itself, we have 2 histogram to study the model.

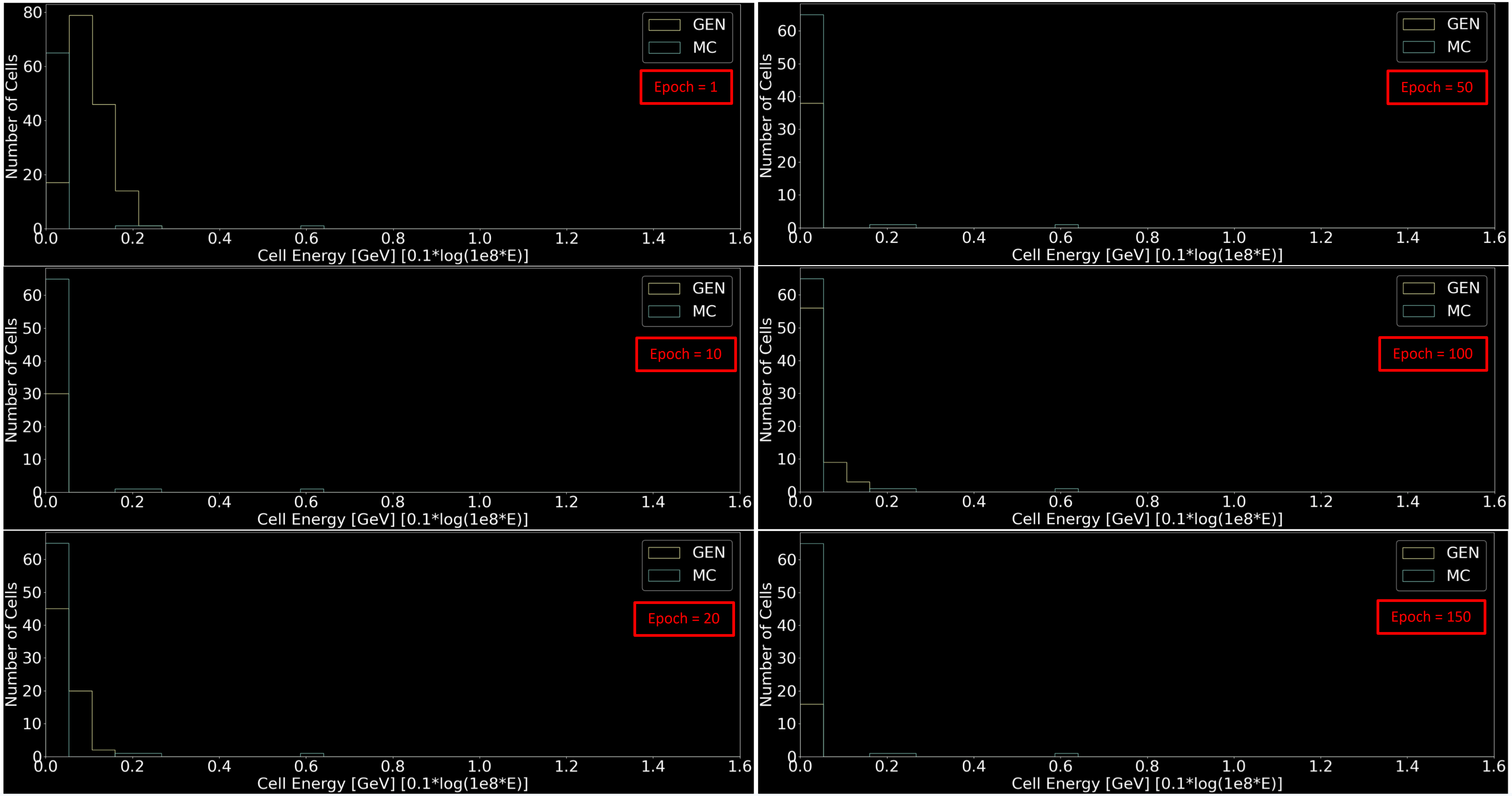


# Test result (image)



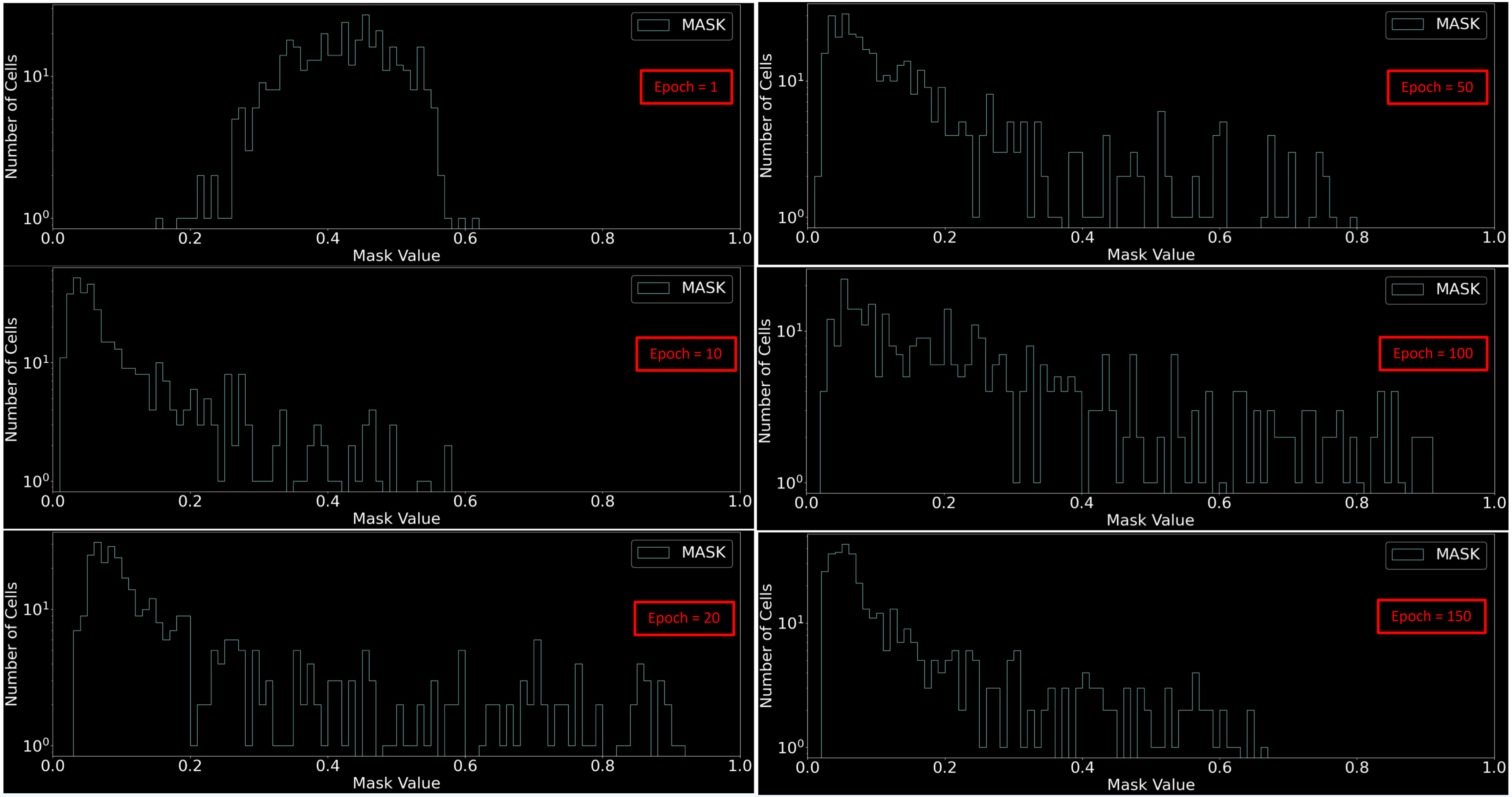
# Test result (hist: MC v GEN cell energy)





# Test result (hist: MC v GEN cell energy)

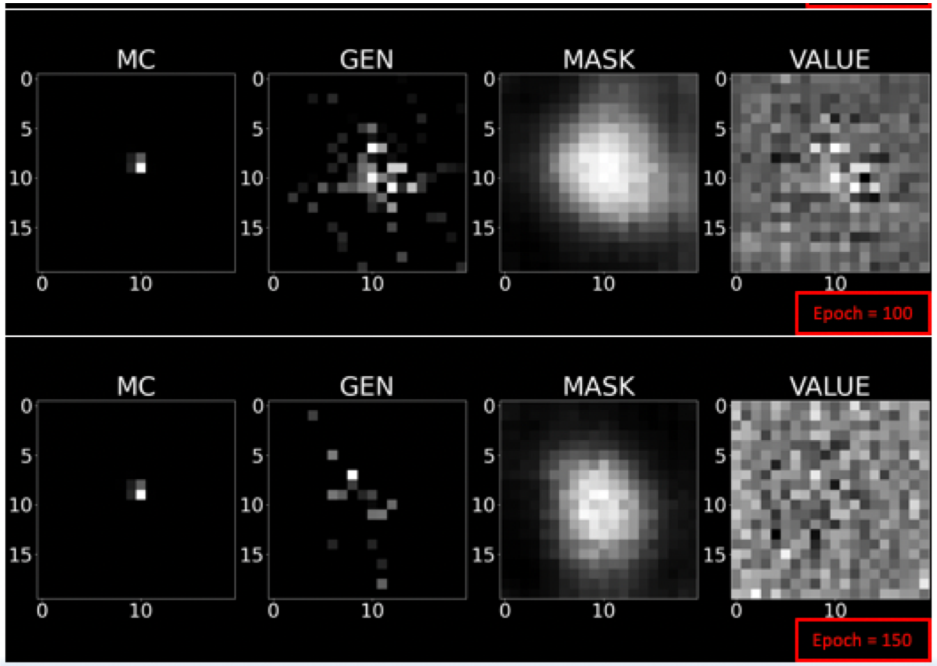


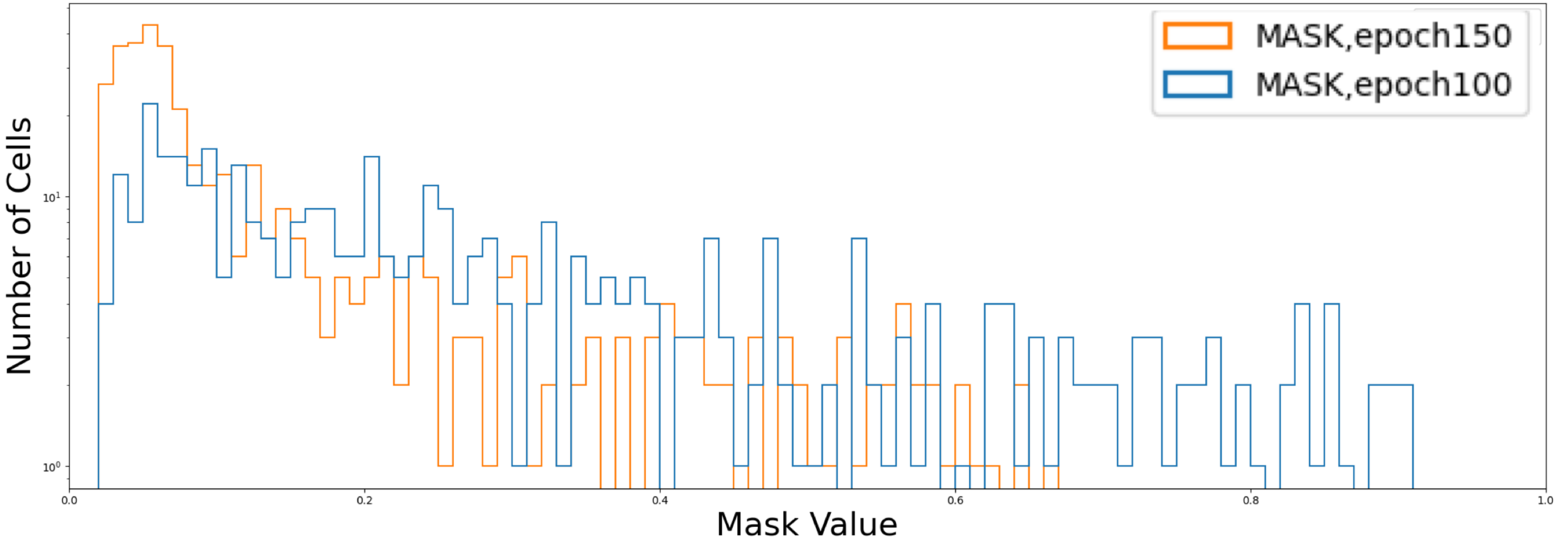


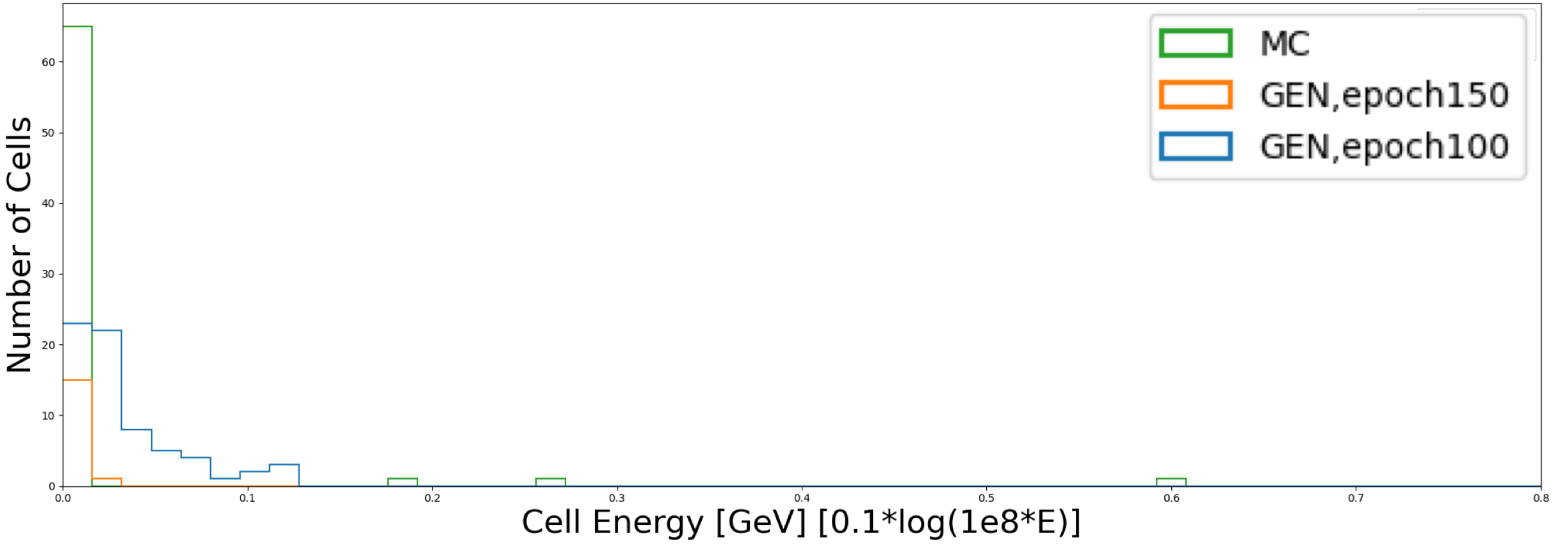
# What do we learn from these plots



- We can see that the model is trying to balance the MC-mimicking and the mask range from the hist.
- Which is very clear if we overlay the hist.



# Summary



- We separated the GAN from the algorithm and now focusing on the training of the regression.
- By checking the image and corresponding histogram, we can see the improvement along the training.
- Now we are training with the same set of parameters with 300 epochs.
- Then we can consider to add weighting to the sum of losses.