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

Status report

2025/06/05

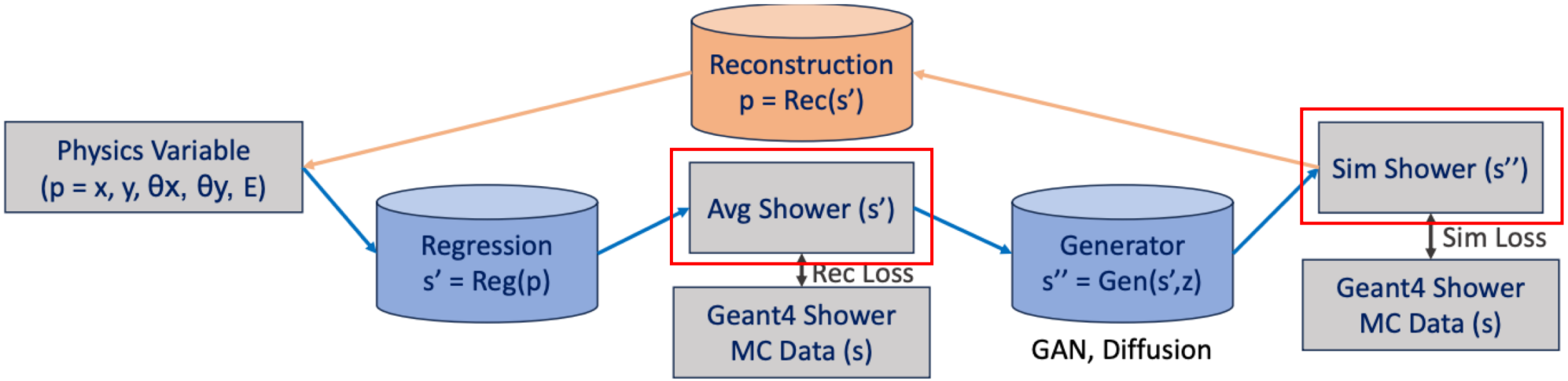
ZDC ML

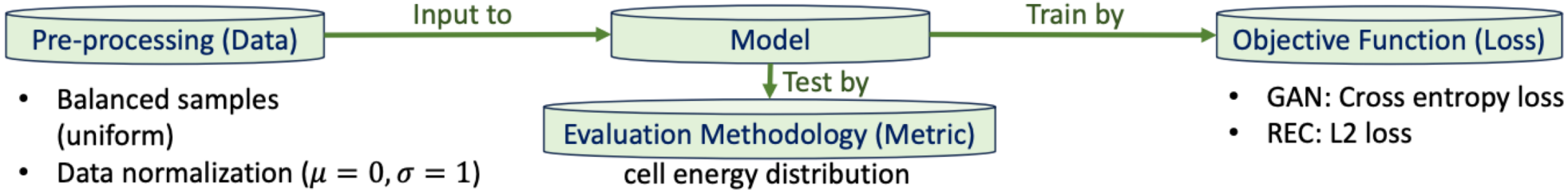
WAI YUEN CHAN

Overview

- Focus on improving the performance of GAN-CNN

Introduction

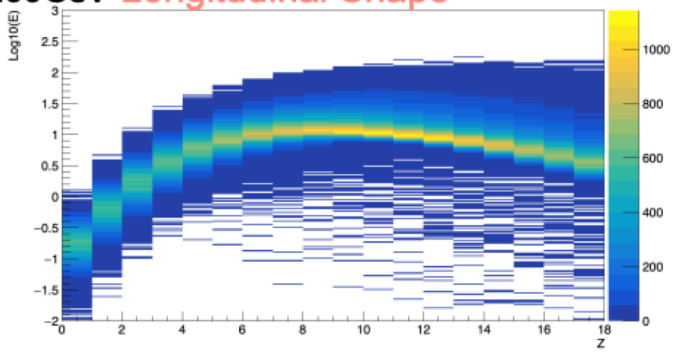




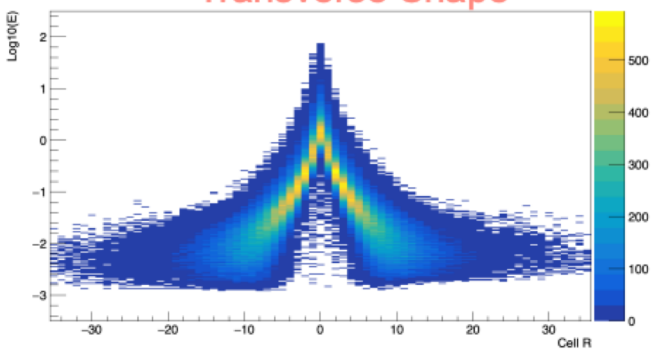
- Balanced samples (uniform)
- Data normalization ($\mu = 0, \sigma = 1$)

- GAN: Cross entropy loss
- REC: L2 loss

e-, 100GeV Longitudinal Shape



Transverse Shape



The longitudinal shower profile is described by the Gamma distribution:

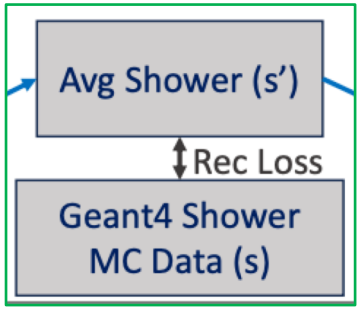
$$\frac{dE}{dt}(t) = E_0 \frac{(\beta t)^{\beta T_0} \beta e^{-\beta t}}{\Gamma(\beta T_0 + 1)}, \quad (1)$$

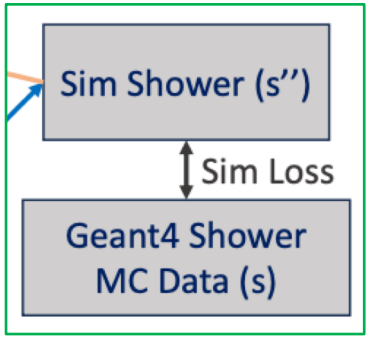
using parameters described above. The scale parameter β is found to be constant over the wide energy range, therefore it is fixed in the present analysis. Individual shower parameters E_0 and T_0 are obtained from a fit to observed energy depositions in the ECAL cells of that shower. Fig. 7 shows the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

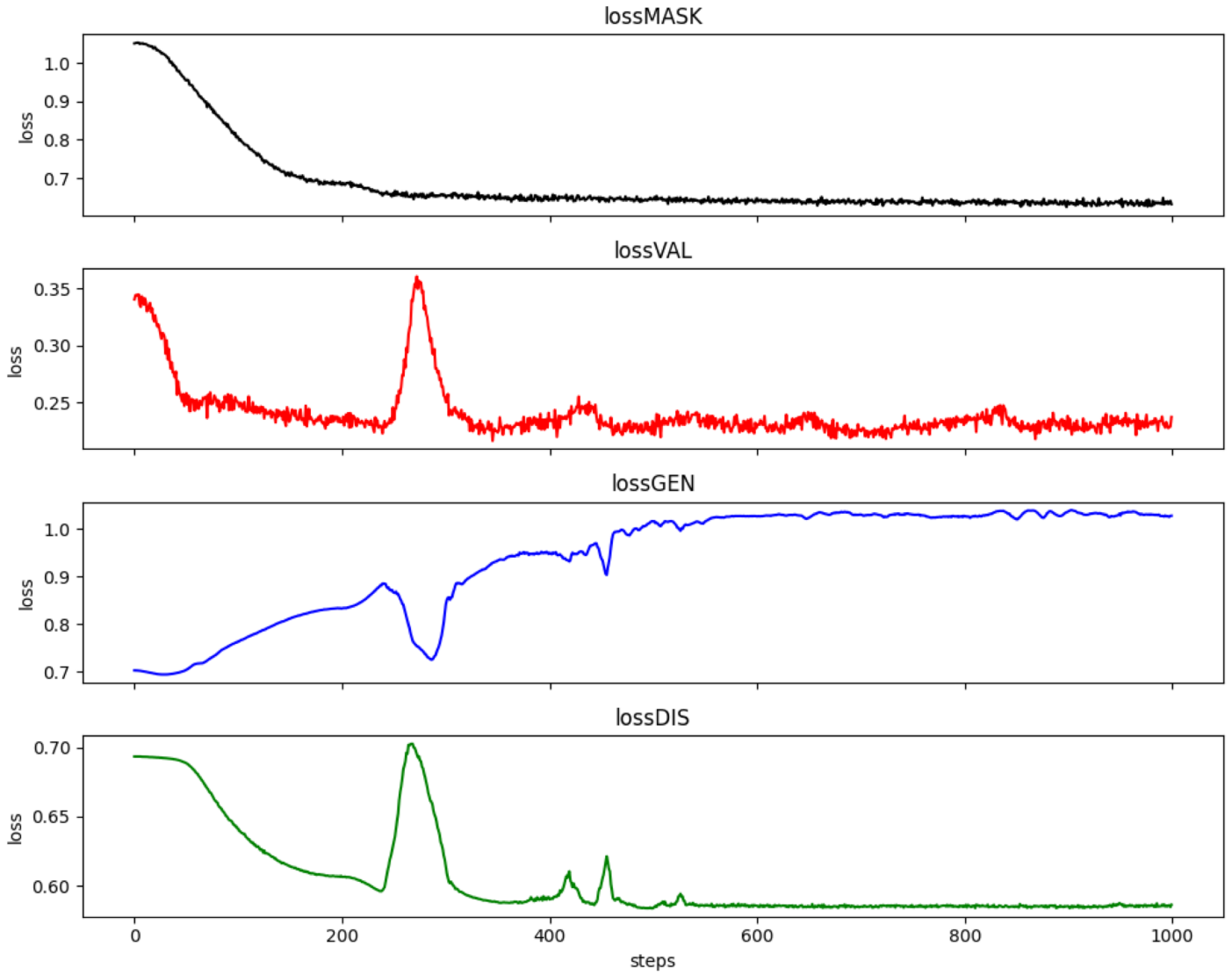
At a given shower depth, the transverse shower shape as a function of the distance from the shower axis r is described by the sum of a narrow core and a wide tail:

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

Recap: Training result





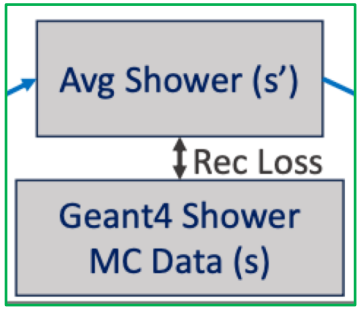


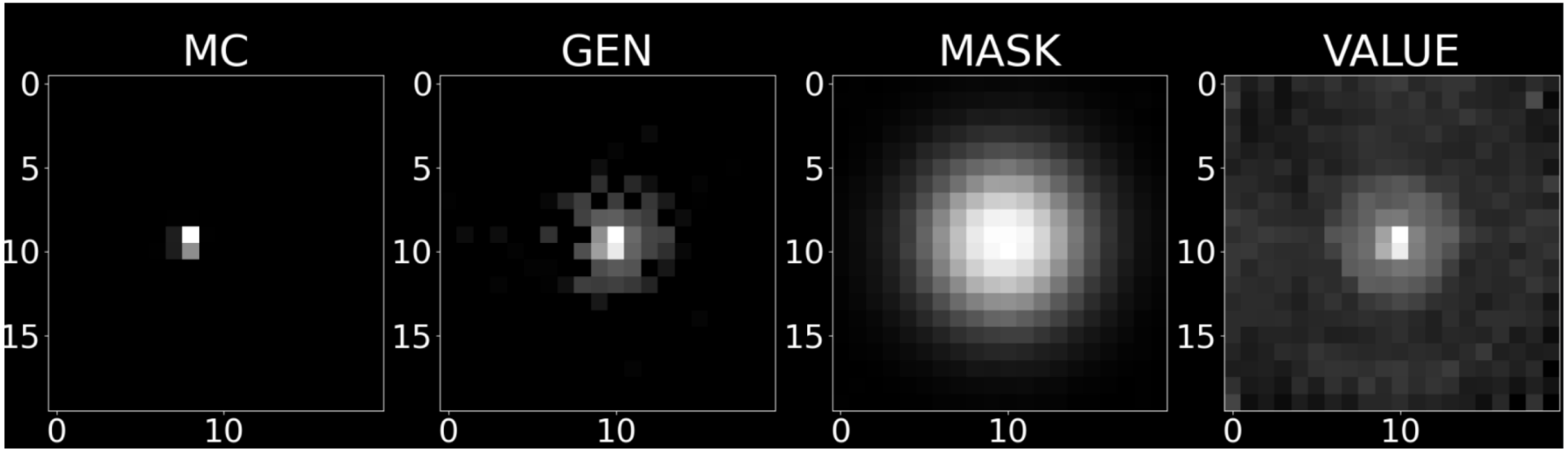
We have 4 losses and corresponding to different parameters.

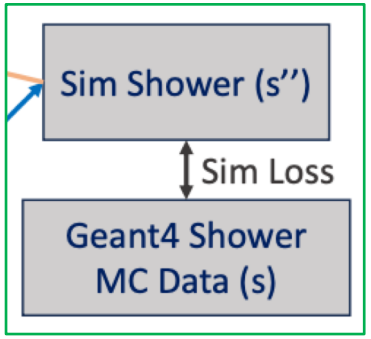
The training use 4000 events and the test use 1000 events in this test run.

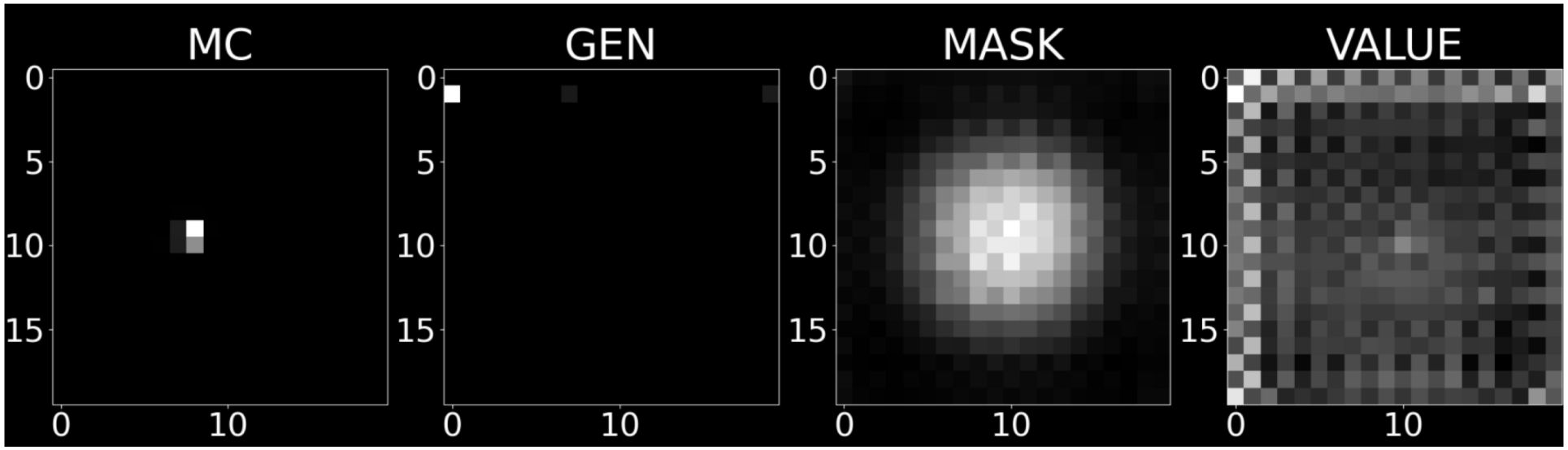
1000 steps = 500 epochs have been used.

Recap: Training result



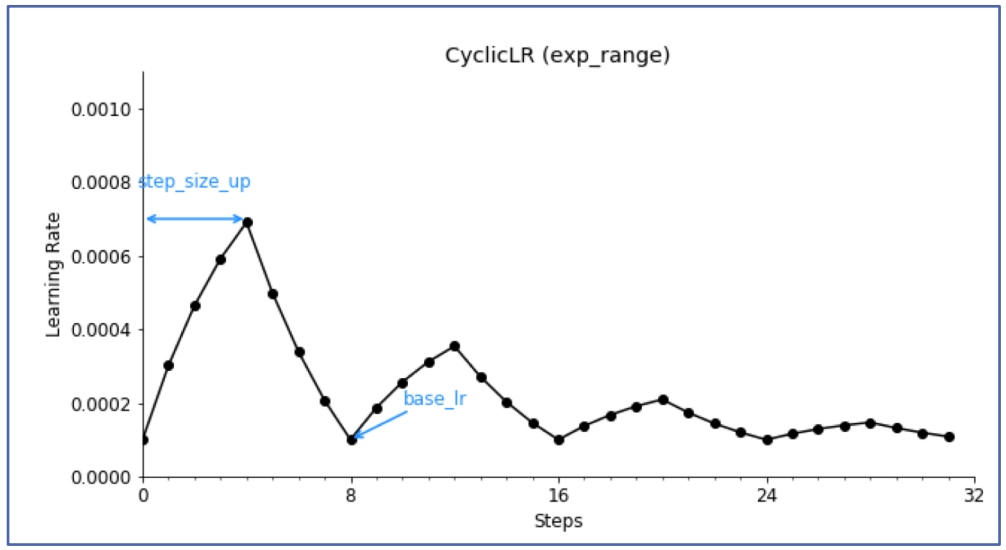




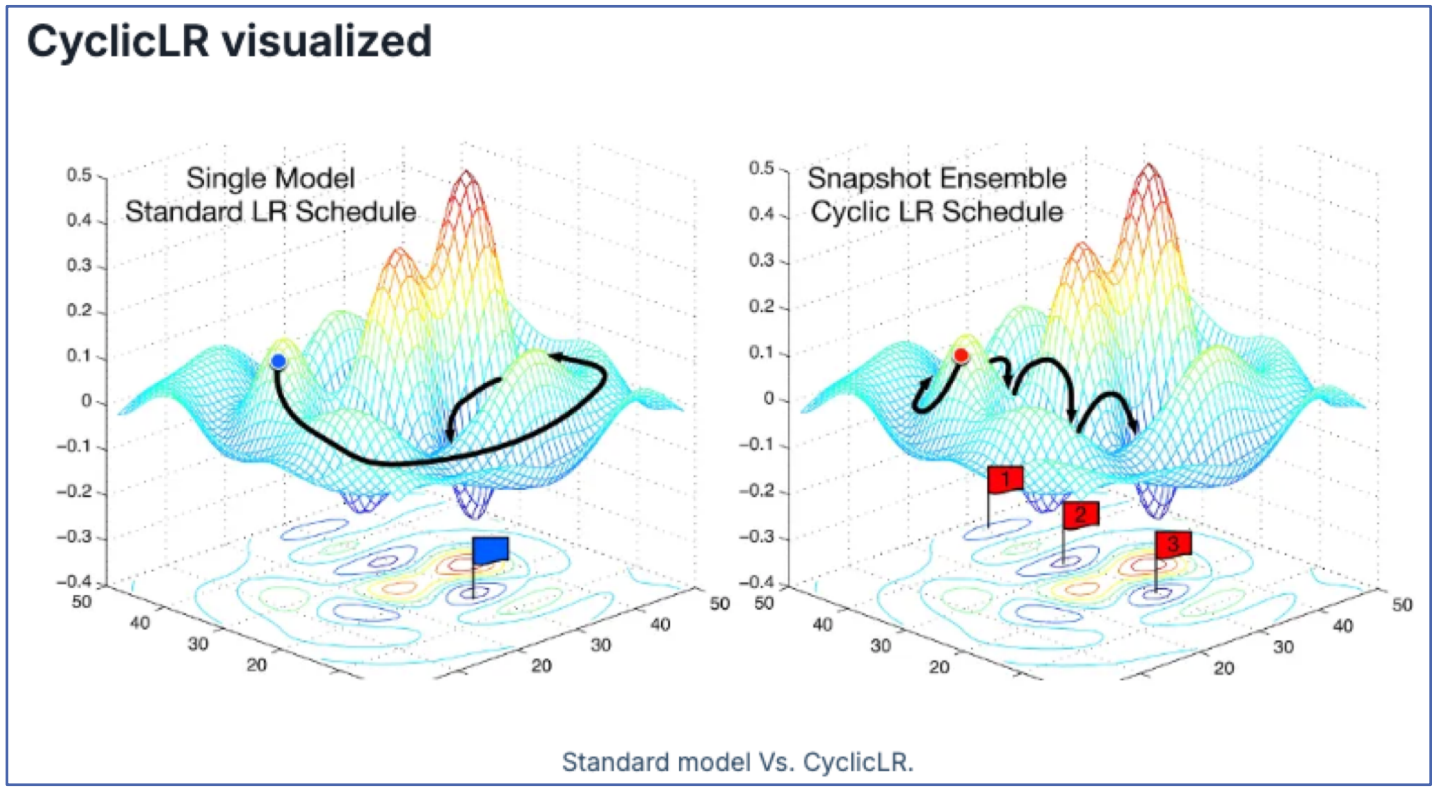


Hyperparameters in the ML training

- Learning rate scheduler: a function which dynamically adjusts the learning rate over the training. The learning rate cycles through these bounds in a triangular or exponential pattern, which helps the model escape local minima and converge faster.
- Here we use ”CyclicLR”




```
cyclic_base_lr = 1e-6
cyclic_max_lr = 1e-4
cyclic_step_size_up = 40
cyclic_step_size_down = 160
cyclic_mode = 'exp_range'
cyclic_gamma = 0.999925
```



Hyperparameters in the ML training

- We have tried different combination of hypermeters, in order to stabilise the loss function.

CyclicLR: default

```
cyclic_base_lr = 1e-6
cyclic_max_lr = 1e-4
cyclic_step_size_up = 40
cyclic_step_size_down = 160
cyclic_mode = 'exp_range'
cyclic_gamma = 0.999925
```

CyclicLR: reduced

```
cyclic_base_lr = 1e-6
cyclic_max_lr = 5e-5
cyclic_step_size_up = 80
cyclic_step_size_down = 80
cyclic_mode = 'exp_range'
cyclic_gamma = 0.999
```

CyclicLR: asymmetry

```
cyclic_base_lr = 1e-6
cyclic_max_lr_GEN = 1e-4 #GEN
cyclic_max_lr_DIS = 7e-5 #DIS
cyclic_step_size_up = 100
cyclic_step_size_down = 100
cyclic_mode = 'exp_range'
cyclic_gamma = 0.995
```

Name	CNN	Label ratio	Training ratio	CyclicLR setting	epochs
Default	default	0.73:0.27	1:1	default	100
Test 1	simplified	0.73:0.27	1:1	default	100
Test 2	simplified	0.8:0.2	1:1	default	150
Test 3	simplified	0.8:0.2	1:1	deduced	150
Test 4	simplified	0.7:0.3	1:1	asymmetry	150
Test 5	simplified	0.65:0.35	1:3	asymmetry	150
Test 6	simplified	0.6:0.4	1:3	asymmetry	150

Hyperparameters in the ML training

- We have tried different combination of hypermeters, in order to stabilise the loss function.

CNN: default (example)

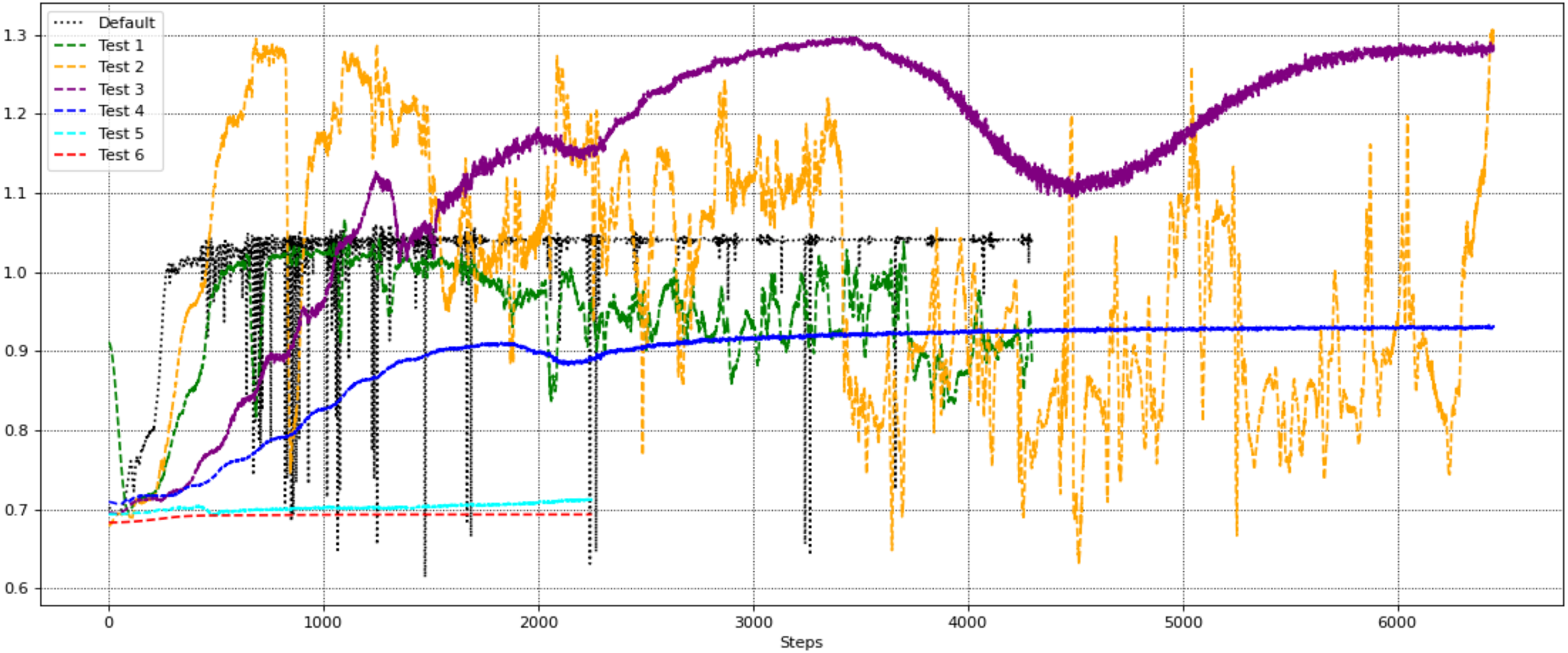
```
self.proj_head = nn.Sequential(
        Conv2dBlockH5W5(hidden_dim, hidden_dim),
        Conv2dBlockH5W5(hidden_dim, hidden_dim),
        PixelUnshuffle2D(2, 2), # (10, 10)
        Conv2dBlockH5W5(hidden_dim*4, hidden_dim*2),
        Conv2dBlockH5W5(hidden_dim*2, hidden_dim),
        PixelUnshuffle2D(2, 2), # (5, 5)
        Conv2dBlockH3W3(hidden_dim*4, hidden_dim*2),
        Conv2dBlockH3W3(hidden_dim*2, hidden_dim),
        nn.Conv2d(hidden_dim, hidden_dim,
                  kernel_size = (3, 3), stride = (1, 1),
                  padding = (0, 0)), # (3, 3)
        nn.Flatten(start_dim = 1, end_dim = 3), # 9 = 3 x 3
        LinearBlock(hidden_dim*9, hidden_dim*4, 4),
        LinearBlock(hidden_dim*4, hidden_dim*4, 4),
        nn.Linear(hidden_dim*4, 1),
        nn.Sigmoid()
        )
```

CNN: simplified (example)

```
self.proj_head = nn.Sequential(
        Conv2dBlockH5W5(hidden_dim, hidden_dim),
        #Conv2dBlockH5W5(hidden_dim, hidden_dim),
        PixelUnshuffle2D(2, 2), # (10, 10)
        Conv2dBlockH5W5(hidden_dim*4, hidden_dim),
        #Conv2dBlockH5W5(hidden_dim*2, hidden_dim),
        PixelUnshuffle2D(2, 2), # (5, 5)
        Conv2dBlockH3W3(hidden_dim*4, hidden_dim),
        #Conv2dBlockH3W3(hidden_dim*2, hidden_dim),
        nn.Conv2d(hidden_dim, hidden_dim,
                  kernel_size = (3, 3), stride = (1, 1),
                  padding = (0, 0)), # (3, 3)
        nn.Flatten(start_dim = 1, end_dim = 3), # 9 = 3 x 3
        LinearBlock(hidden_dim*9, hidden_dim*4, 4),
        #LinearBlock(hidden_dim*4, hidden_dim*4, 4),
        nn.Linear(hidden_dim*4, 1),
        nn.Sigmoid()
        )
```

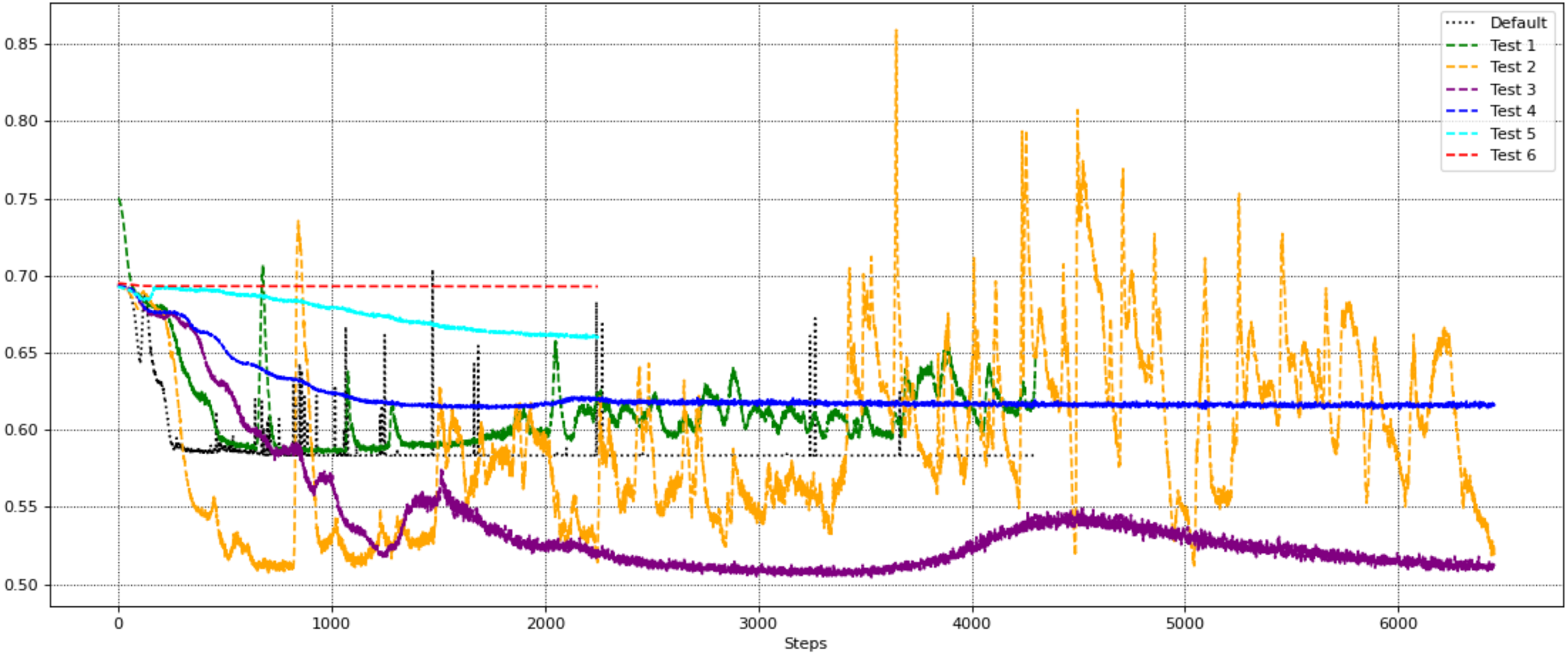
Hyperparameters in the ML training

lossGEN



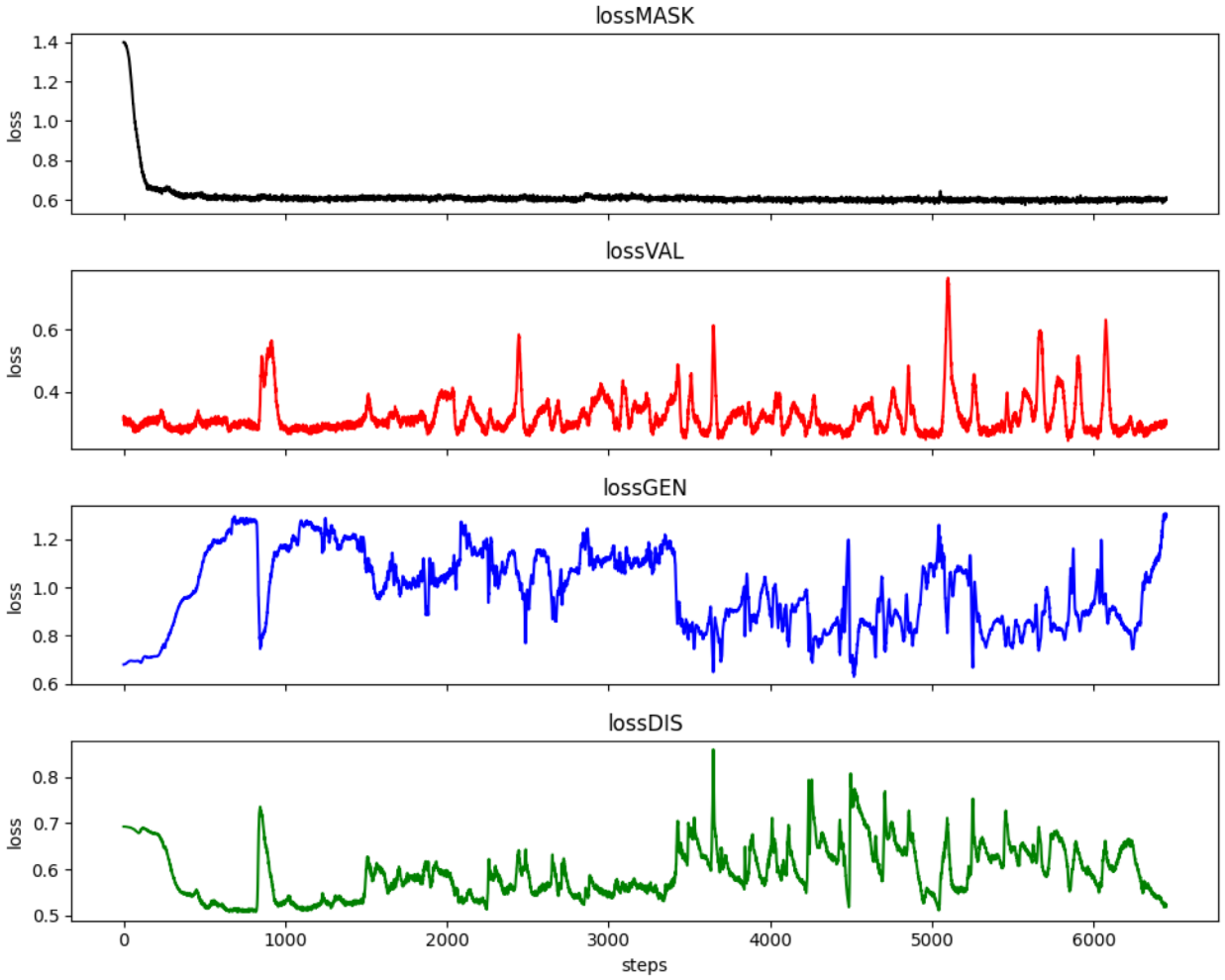
Hyperparameters in the ML training

lossDIS

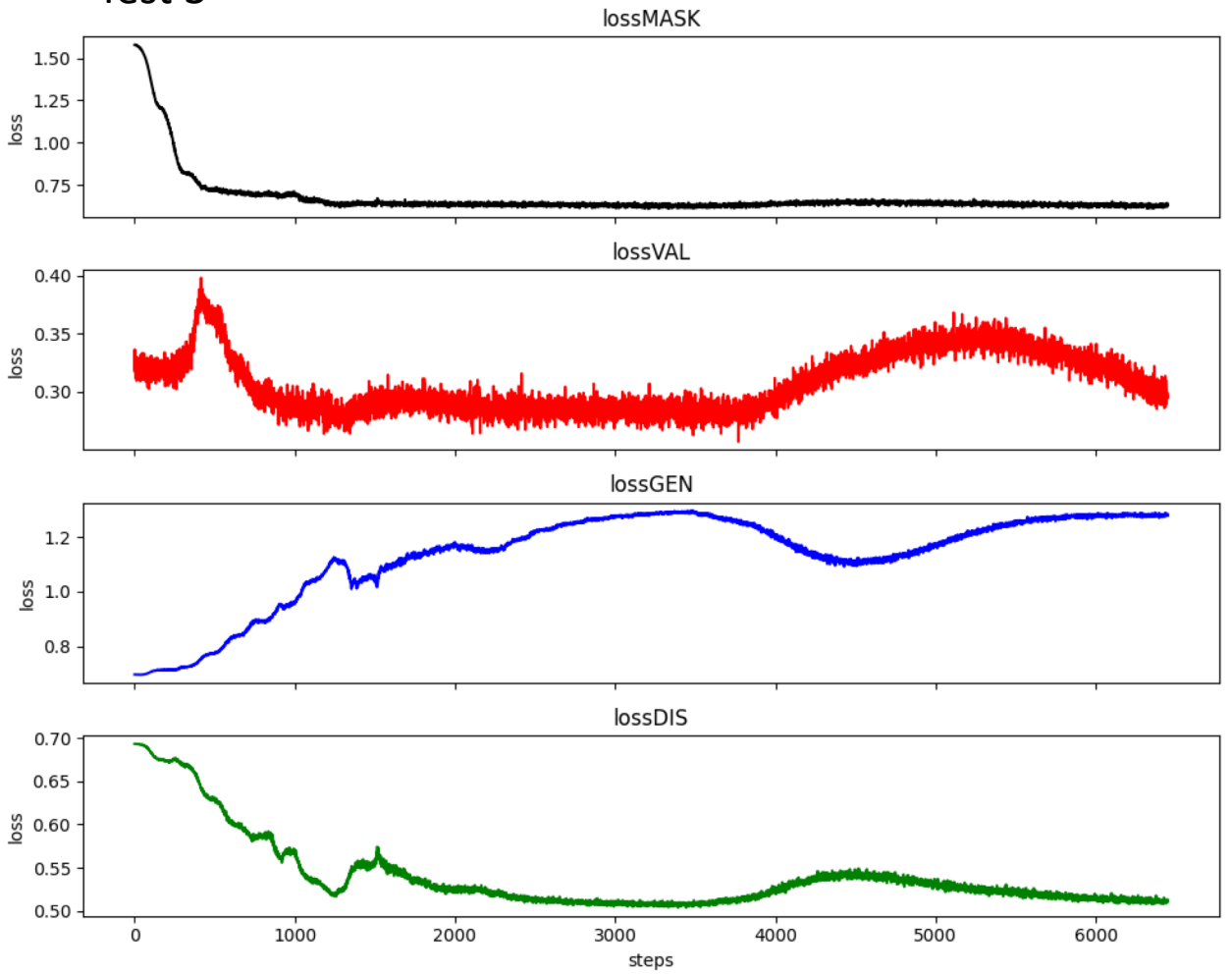


Hyperparameters in the ML training

Test 2

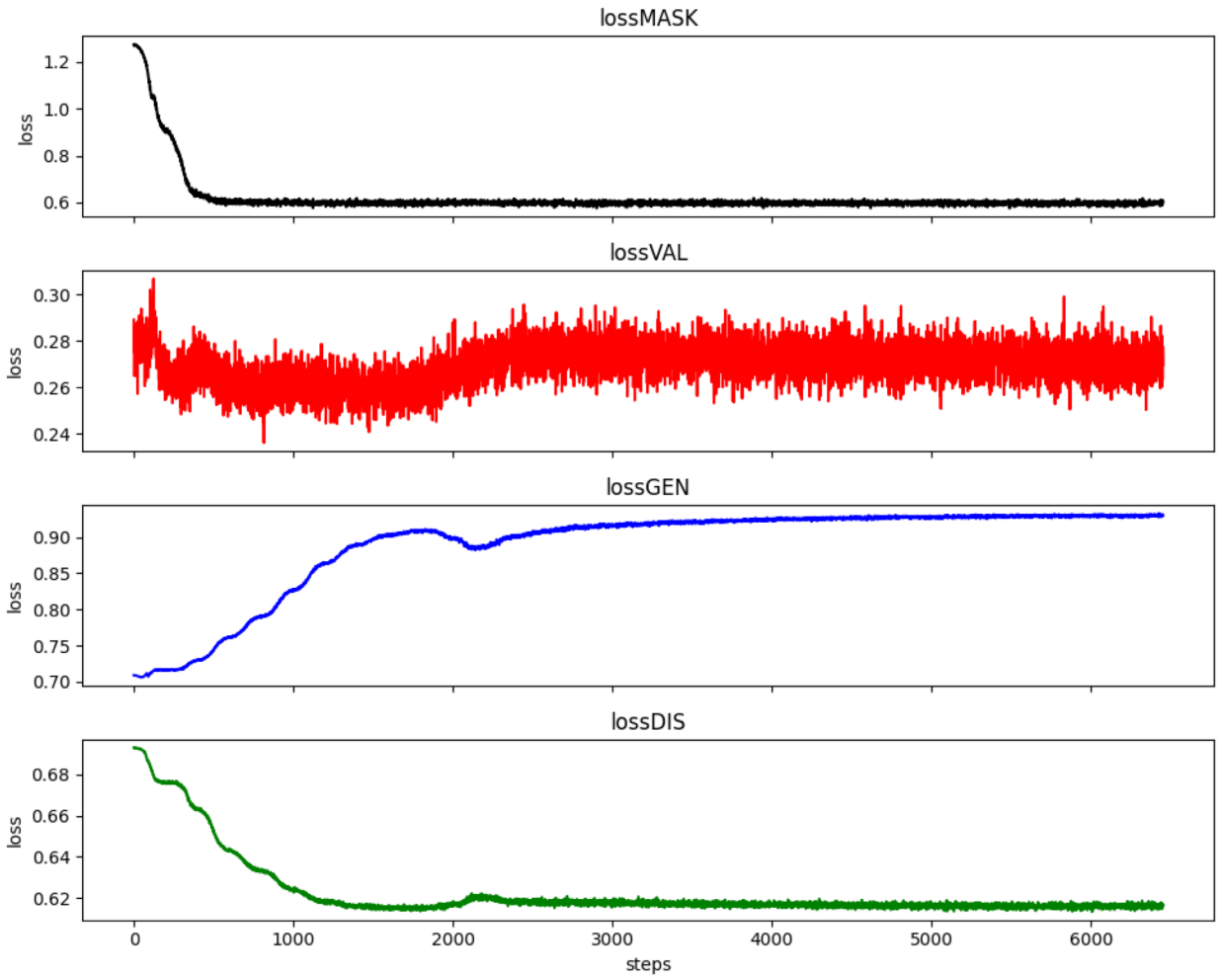


Test 3

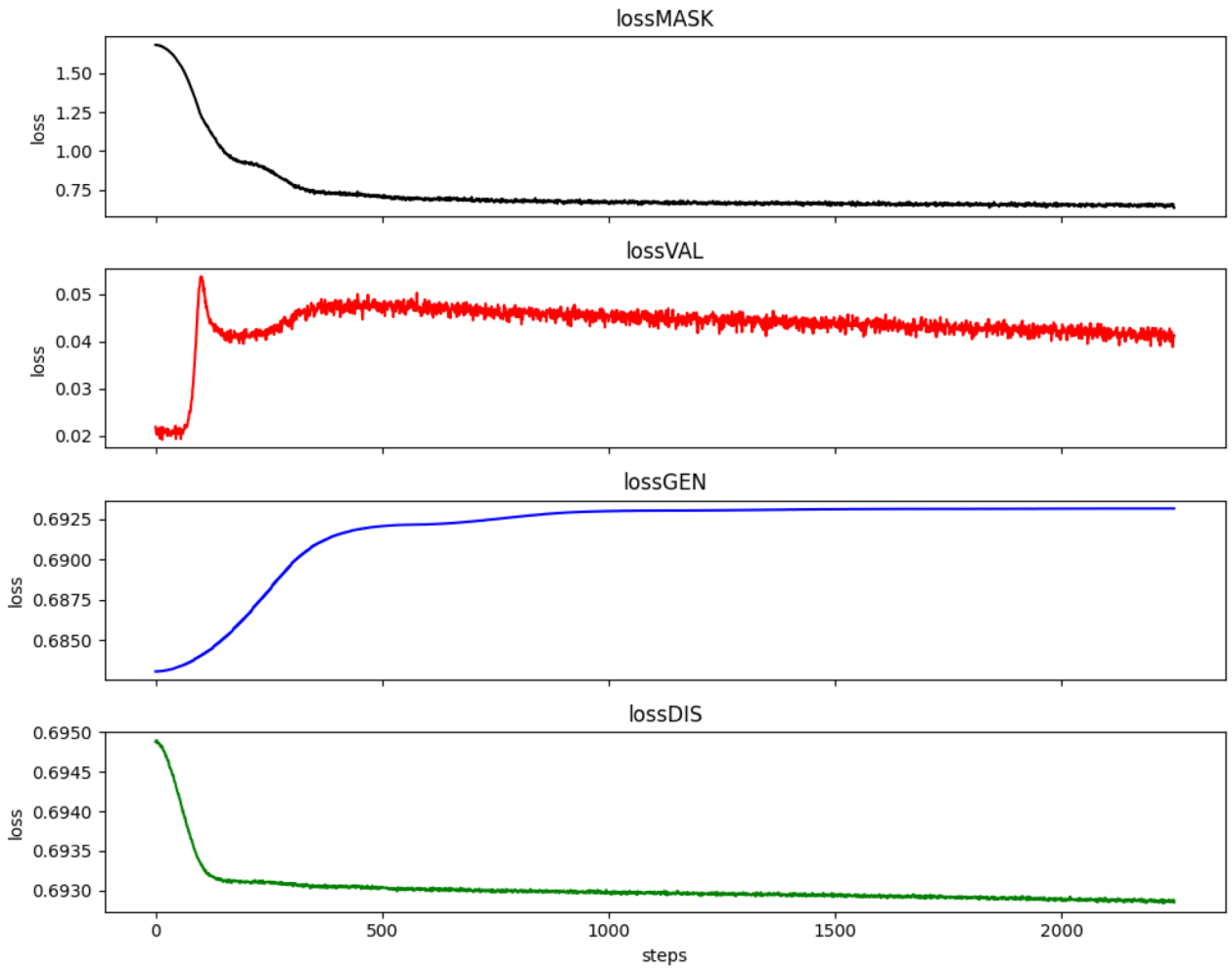


Hyperparameters in the ML training

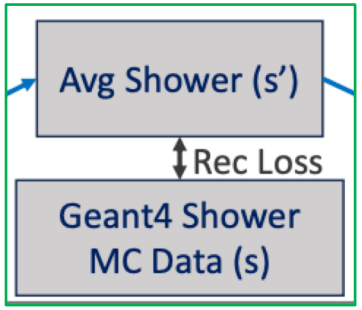
Test 4

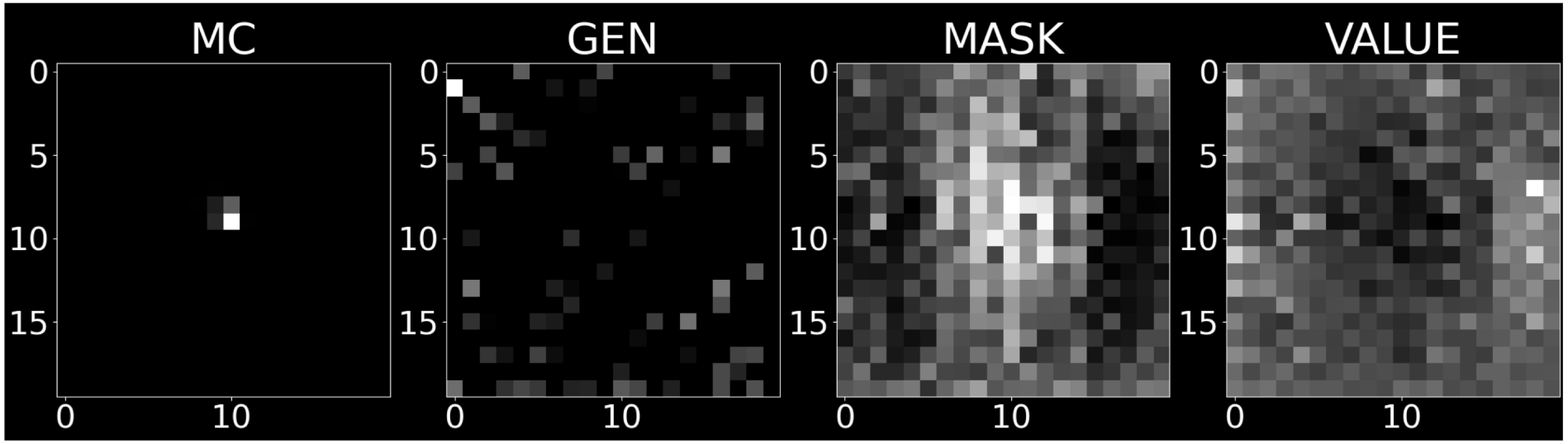


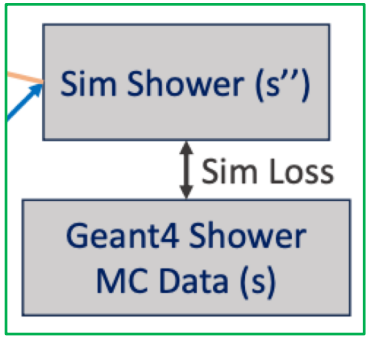
Test 6

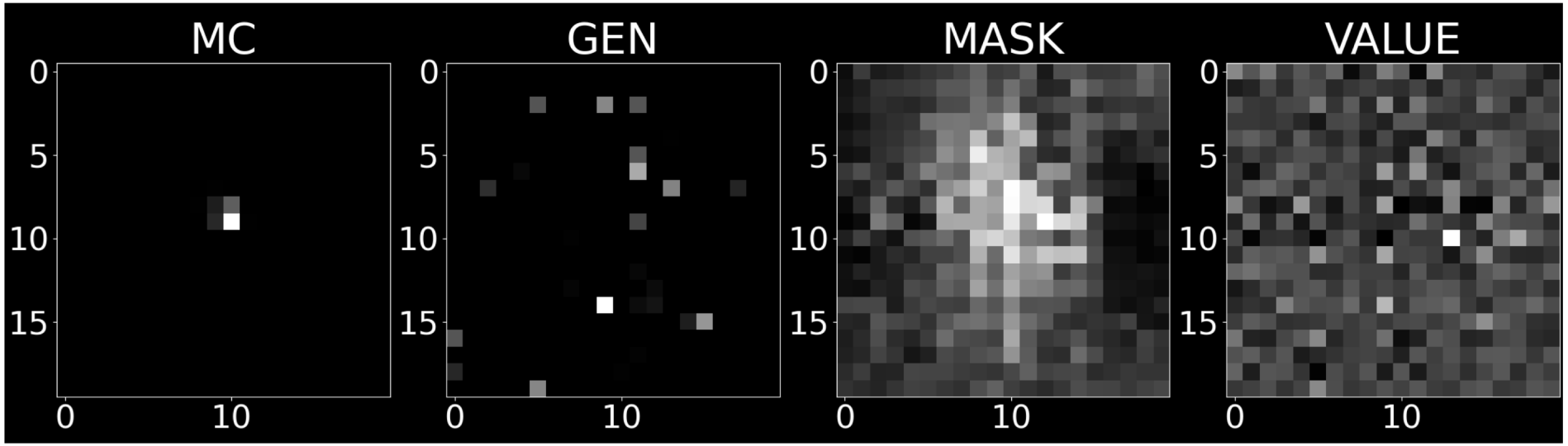


Test result from Test 6 model





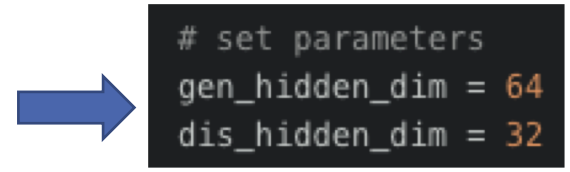




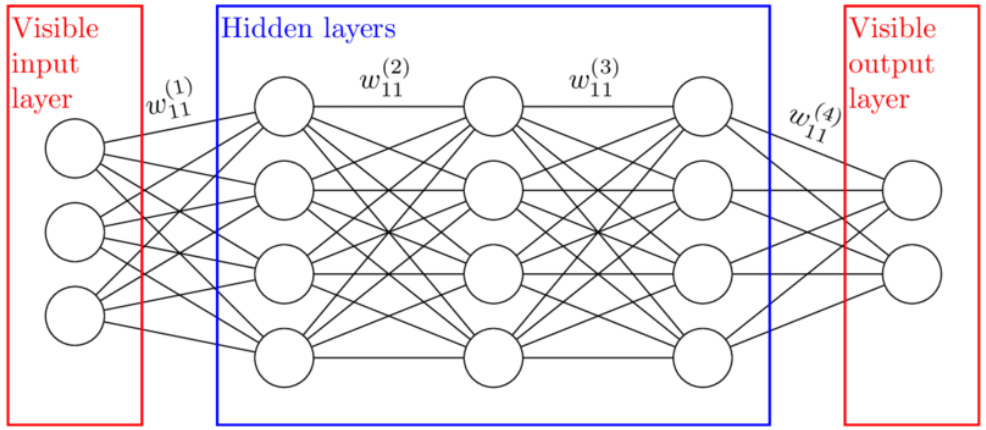
Hyperparameters (II)

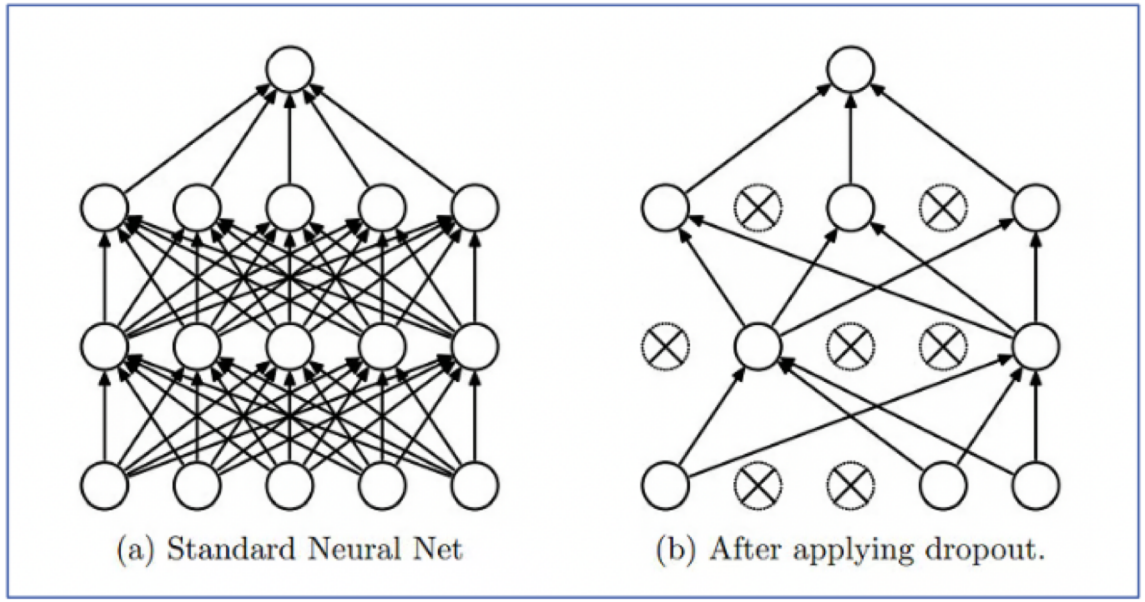
- We have tried more different combination of hypermeters but still can't fix the generated image.
- Most recent attempt: Extend the hidden dim (Number of neurons) of the network with dropout layers with dropout rate = 10%.
- Also, bring the label ratio into 0.8:0.2; using default CNN structure.

```
gen_hidden_dim = 16
dis_hidden_dim = 8
```



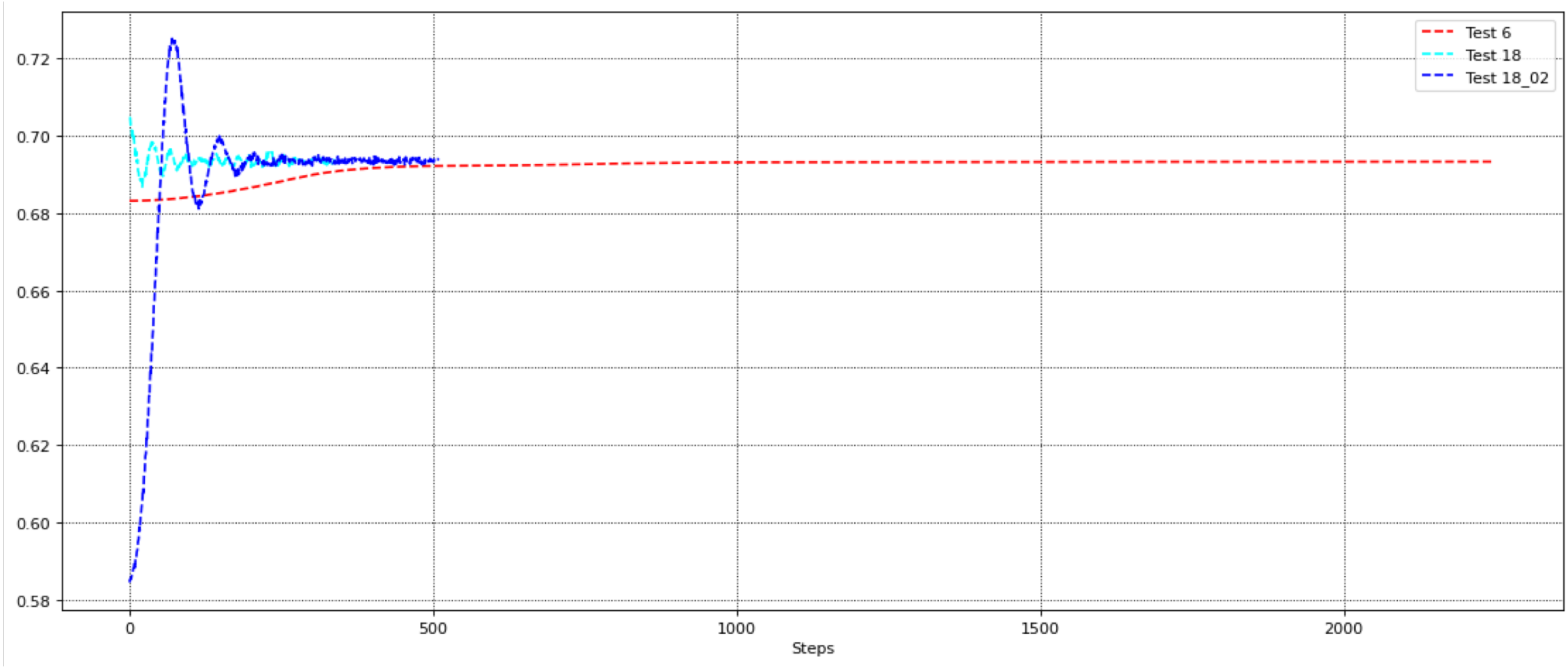
```
# set parameters
gen_hidden_dim = 64
dis_hidden_dim = 32
```





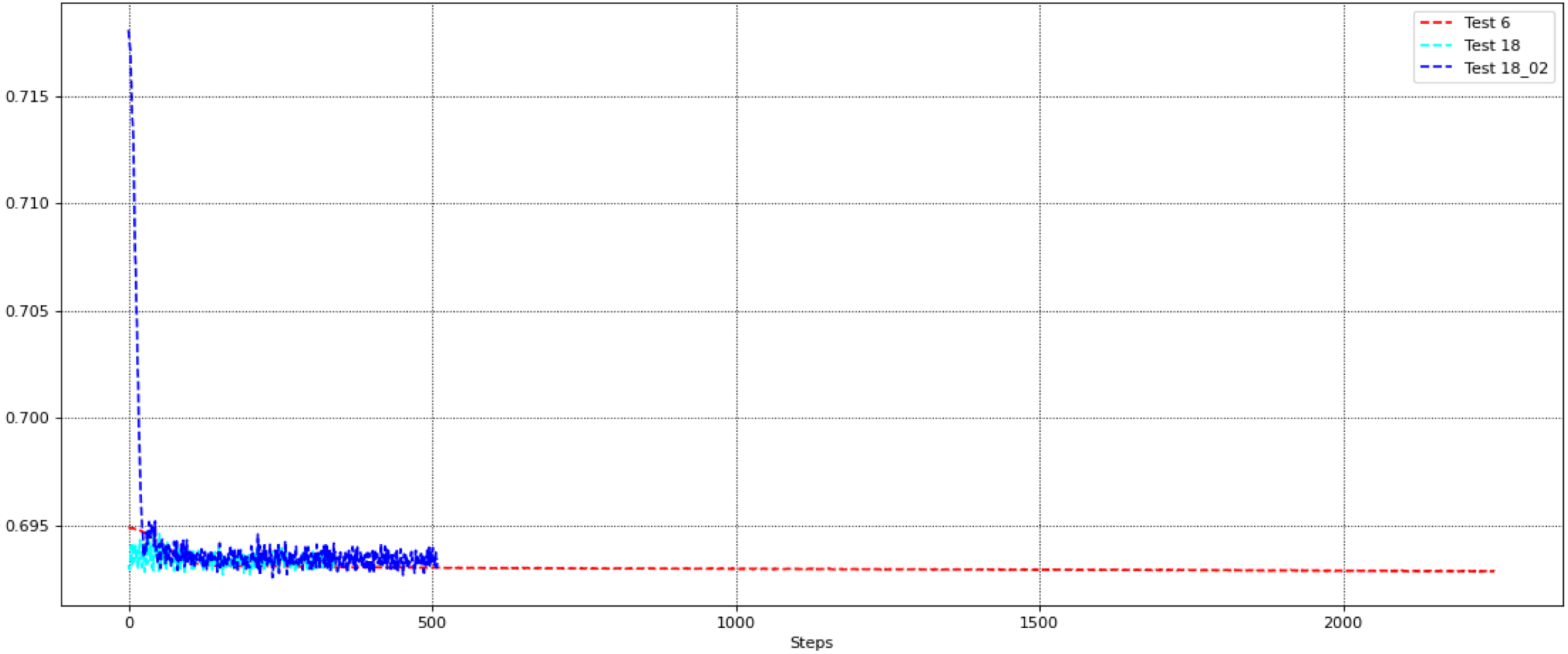
Hyperparameters in the ML training

lossGEN



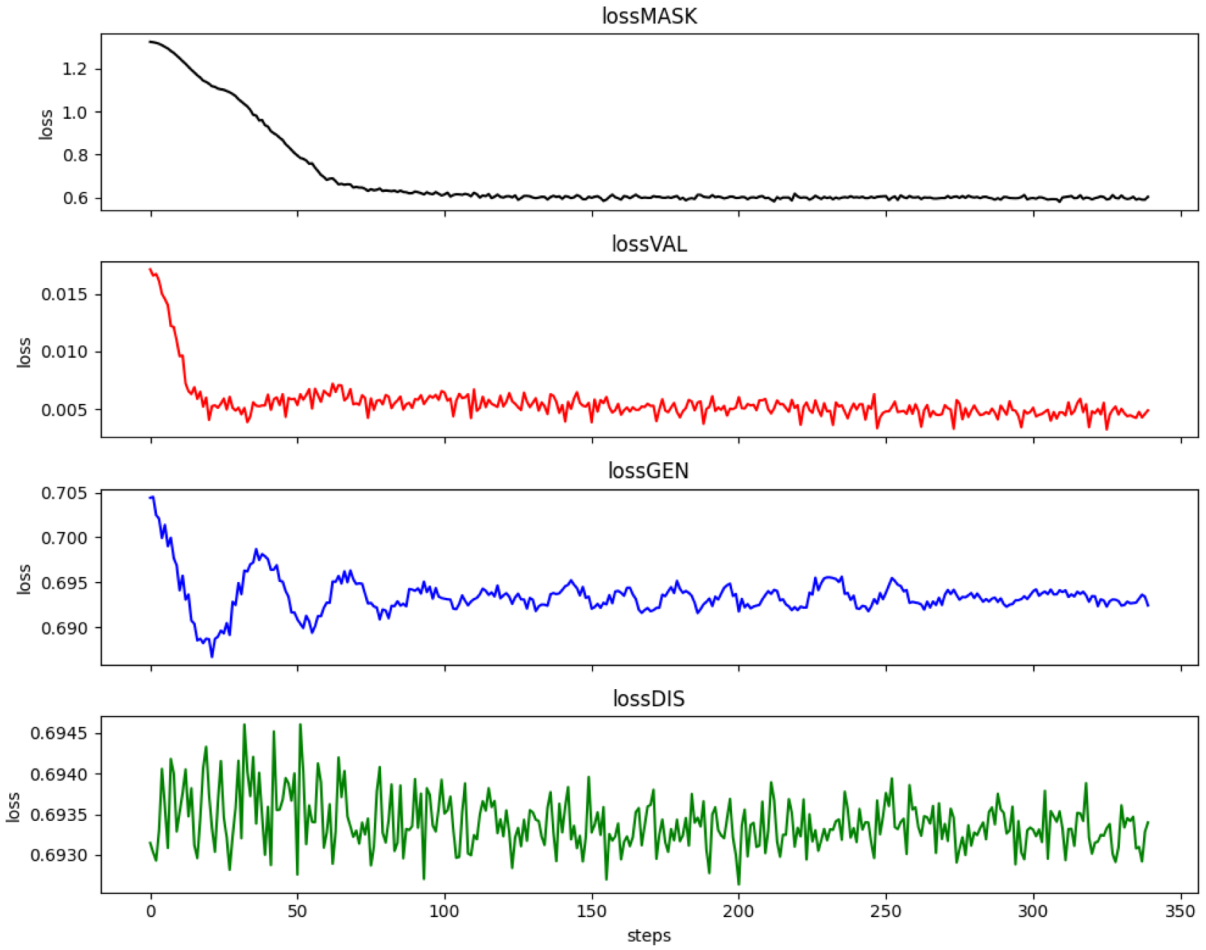
Hyperparameters in the ML training

lossDIS

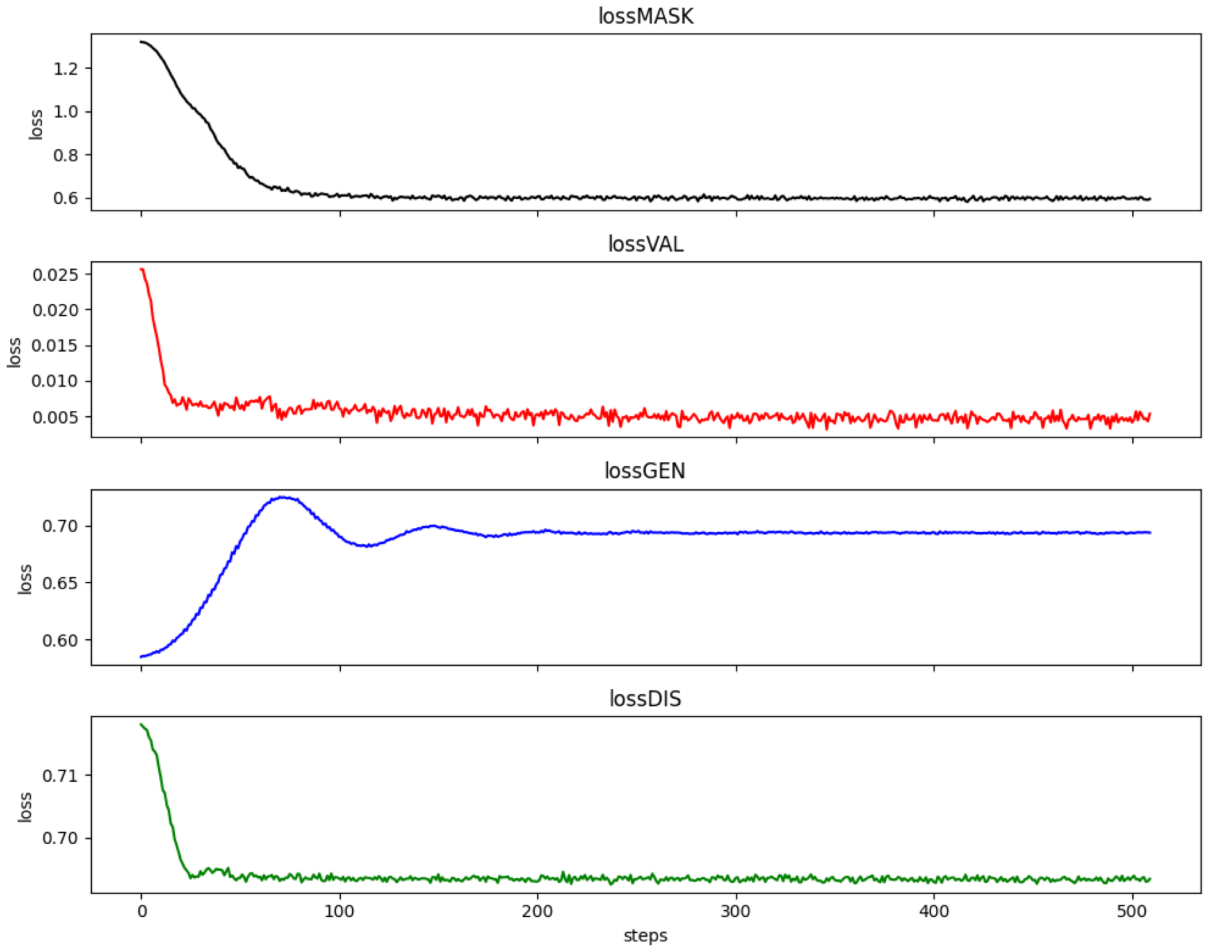


Hyperparameters in the ML training

Test 18

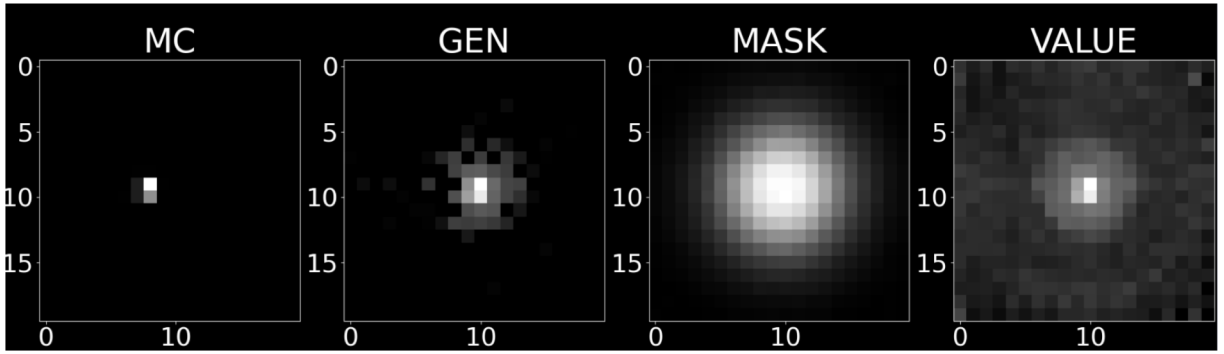


Test 18_02

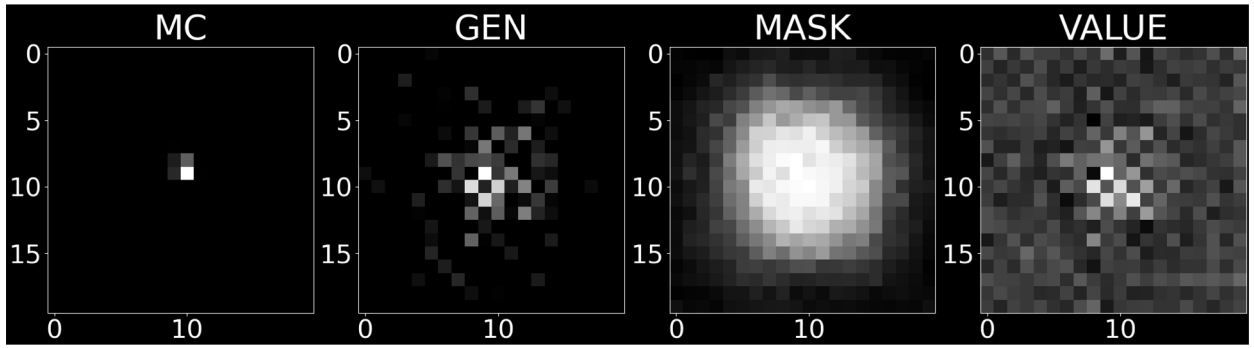


Test result (Gen)

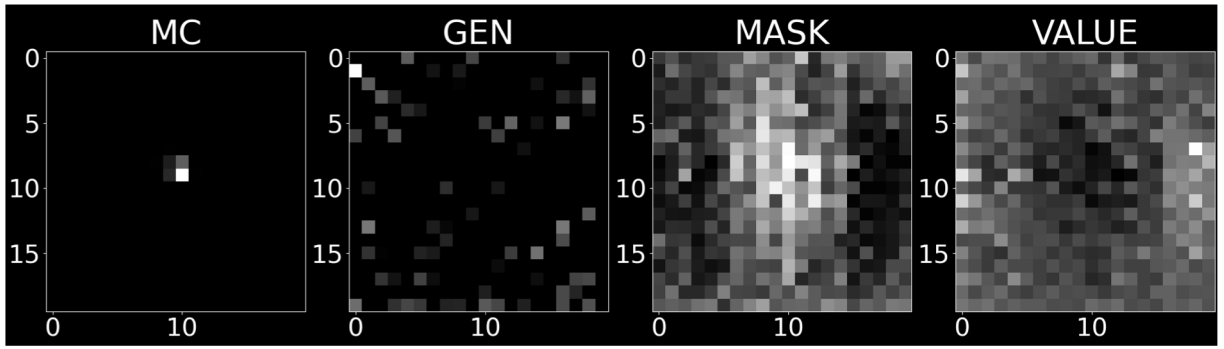
Default



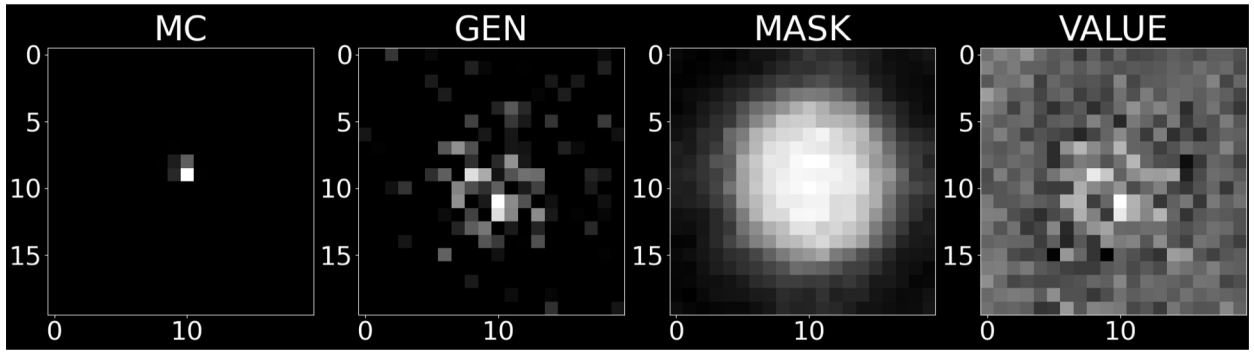
Test 18



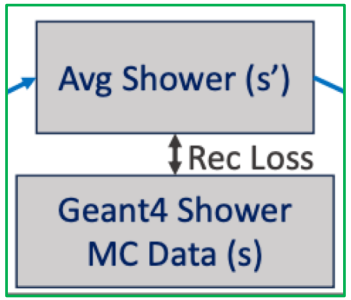
Test 6



Test 18_02

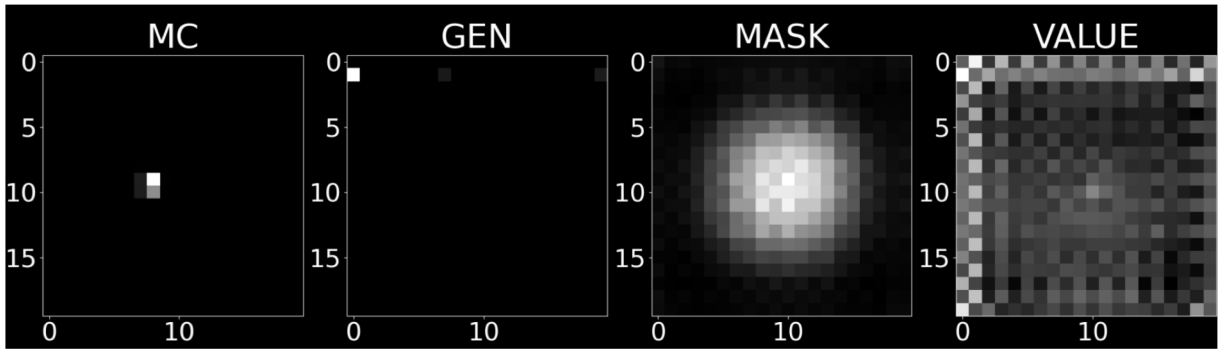


- Looks like we fixed the image in GEN, but still worse than the default image.

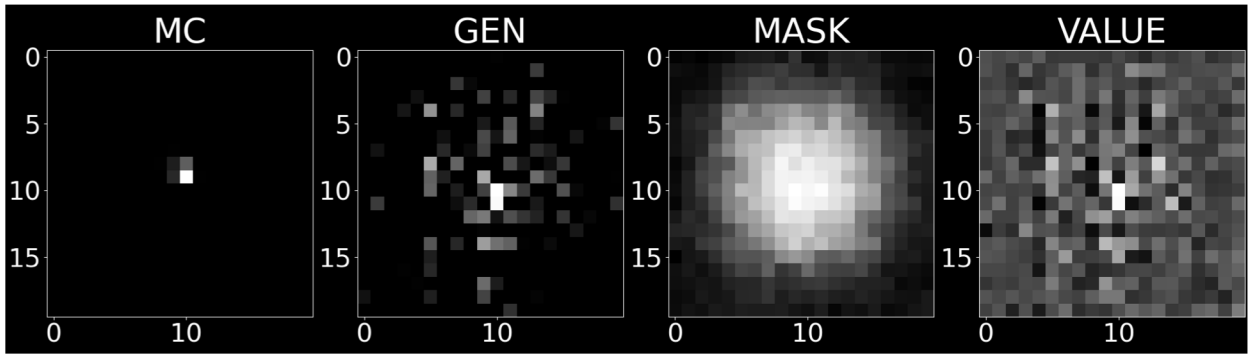


Test result (Sim)

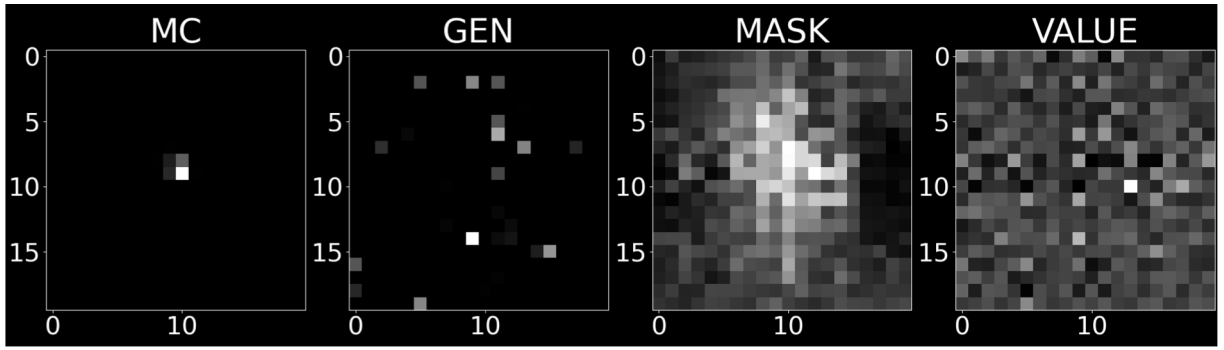
Default



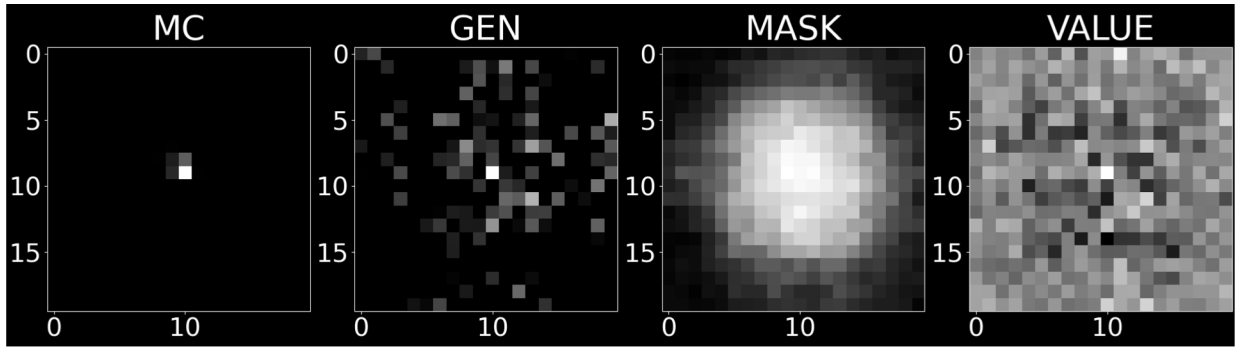
Test 18



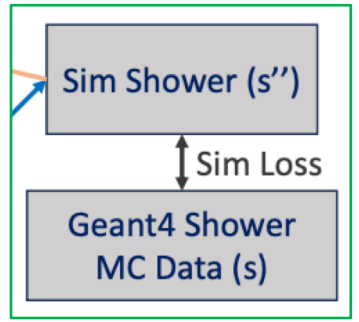
Test 6



Test 18_02



- Looks like we have a better MASK iamge compare to test 6, and more spots in GEN compare to default.



Summary & Outlook

- Looks like the default structure give the best image in GEN so far, but the discriminator need more adjustment.
- We start from there and fine-tuned the hyperparameters and eventually have a model that always give ideal lossGEN and lossDIS.
- However, the generated image become noise-like, doesn't looks like the MC image at all.
- Although the most recent attempt can somehow bring it back but we don't have that much computing resource to extend the network further.
- We can either:
  - Figure out a way to extend the network further without using up the computing resource
  - Keep tuning the hyperparameter to optimize the model and give a reasonable resolution in the generator with a reliable discriminator
  - Go back to the default setting and find a new direction to optimize the model