

**Problem 1**

1. Print the network architecture of your model.

```
ResNet(  
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)  
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
  (relu): ReLU(inplace=True)  
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)  
  (layer1): Sequential(  
    (0): BasicBlock(  
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)  
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
      (relu): ReLU(inplace=True)  
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)  
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    )  
    (1): BasicBlock(  
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)  
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
      (relu): ReLU(inplace=True)  
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)  
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    )  
    (2): BasicBlock(  
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)  
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
      (relu): ReLU(inplace=True)  
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)  
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    )  
  )  
  (layer2): Sequential(  
    (0): BasicBlock(  
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)  
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```

(relu): ReLU(inplace=True)
(conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(downsample): Sequential(
  (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
  (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
)
(1): BasicBlock(
  (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(2): BasicBlock(
  (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(3): BasicBlock(
  (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
)
(layer3): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (downsample): Sequential(

```

```

    (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(1): BasicBlock(
  (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(2): BasicBlock(
  (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(3): BasicBlock(
  (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(4): BasicBlock(
  (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)
(5): BasicBlock(
  (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
)

```

```

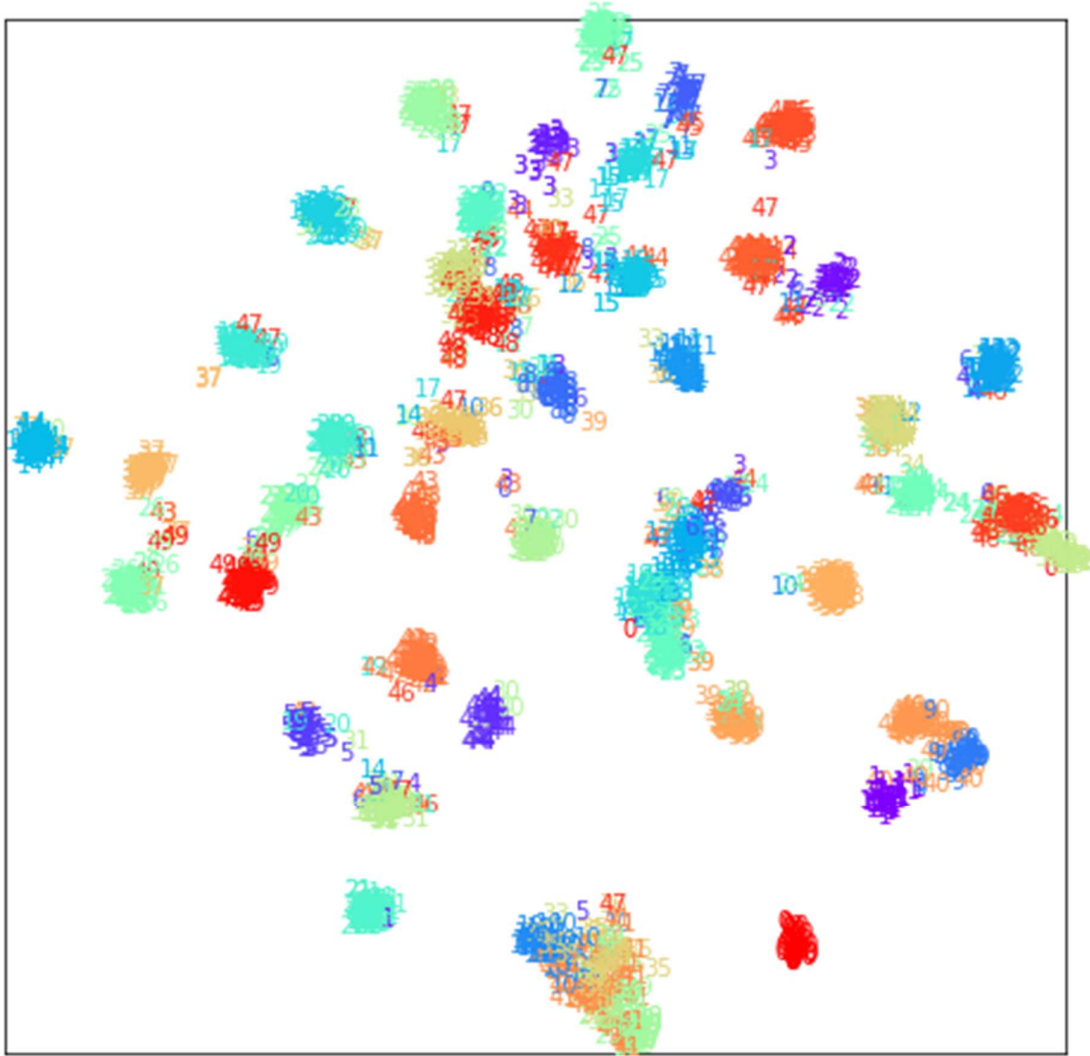
    )
)
(layer4): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
)
  (1): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (2): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=50, bias=True)
)

```

2. Report accuracy of model on the validation set.

0.86346

3. Visualize the classification result on validation set by implementing t-SNE on output features of the second last layer. Briefly explain your result of the tSNE visualization.
- 從結果可看出同樣類別的大致會聚在一起，可幫助最後一層預測分類，也代表 model 有學到如何分類，而有些顏色會混在一起、沒有分得很開(例如最下面藍色、橘色、綠色、土黃色混在一起的區塊)，可能會造成 model 分類錯誤，成為 loss 的來源。



## Problem 2

1. Print the network architecture of your VGG16-FCN32s model.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 512, 512]	1,792
ReLU-2	[-1, 64, 512, 512]	0
Conv2d-3	[-1, 64, 512, 512]	36,928
ReLU-4	[-1, 64, 512, 512]	0
MaxPool2d-5	[-1, 64, 256, 256]	0
Conv2d-6	[-1, 128, 256, 256]	73,856
ReLU-7	[-1, 128, 256, 256]	0
Conv2d-8	[-1, 128, 256, 256]	147,584
ReLU-9	[-1, 128, 256, 256]	0
MaxPool2d-10	[-1, 128, 128, 128]	0
Conv2d-11	[-1, 256, 128, 128]	295,168
ReLU-12	[-1, 256, 128, 128]	0
Conv2d-13	[-1, 256, 128, 128]	590,080
ReLU-14	[-1, 256, 128, 128]	0
Conv2d-15	[-1, 256, 128, 128]	590,080
ReLU-16	[-1, 256, 128, 128]	0
MaxPool2d-17	[-1, 256, 64, 64]	0
Conv2d-18	[-1, 512, 64, 64]	1,180,160
ReLU-19	[-1, 512, 64, 64]	0
Conv2d-20	[-1, 512, 64, 64]	2,359,808
ReLU-21	[-1, 512, 64, 64]	0
Conv2d-22	[-1, 512, 64, 64]	2,359,808
ReLU-23	[-1, 512, 64, 64]	0
MaxPool2d-24	[-1, 512, 32, 32]	0
Conv2d-25	[-1, 512, 32, 32]	2,359,808
ReLU-26	[-1, 512, 32, 32]	0
Conv2d-27	[-1, 512, 32, 32]	2,359,808
ReLU-28	[-1, 512, 32, 32]	0
Conv2d-29	[-1, 512, 32, 32]	2,359,808
ReLU-30	[-1, 512, 32, 32]	0
MaxPool2d-31	[-1, 512, 16, 16]	0
ConvTranspose2d-32	[-1, 512, 32, 32]	2,359,808
ReLU-33	[-1, 512, 32, 32]	0
BatchNorm2d-34	[-1, 512, 32, 32]	1,024
ConvTranspose2d-35	[-1, 256, 64, 64]	1,179,904
ReLU-36	[-1, 256, 64, 64]	0
BatchNorm2d-37	[-1, 256, 64, 64]	512
ConvTranspose2d-38	[-1, 128, 128, 128]	295,040
ReLU-39	[-1, 128, 128, 128]	0
BatchNorm2d-40	[-1, 128, 128, 128]	256
ConvTranspose2d-41	[-1, 64, 256, 256]	73,792
ReLU-42	[-1, 64, 256, 256]	0
BatchNorm2d-43	[-1, 64, 256, 256]	128
ConvTranspose2d-44	[-1, 32, 512, 512]	18,464
ReLU-45	[-1, 32, 512, 512]	0
BatchNorm2d-46	[-1, 32, 512, 512]	64
Conv2d-47	[-1, 7, 512, 512]	231

- Implement an improved model which performs better than your baseline model. Print the network architecture of this model.

VGG-FCN8s

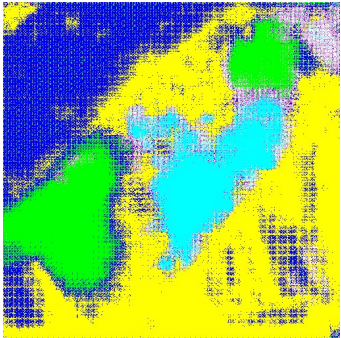
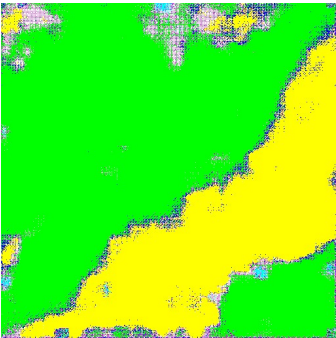

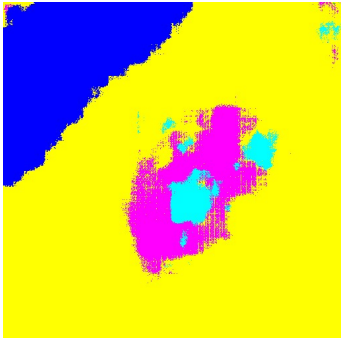
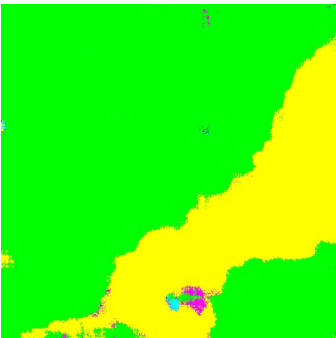
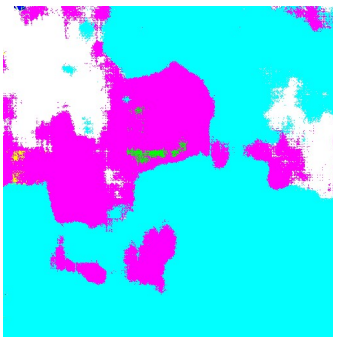

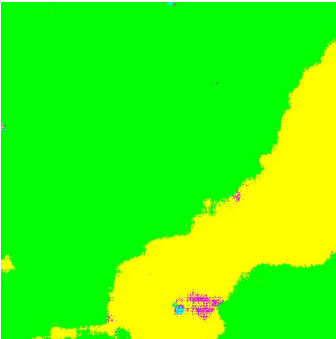

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 512, 512]	1, 792
ReLU-2	[-1, 64, 512, 512]	0
Conv2d-3	[-1, 64, 512, 512]	36, 928
ReLU-4	[-1, 64, 512, 512]	0
MaxPool2d-5	[-1, 64, 256, 256]	0
Conv2d-6	[-1, 128, 256, 256]	73, 856
ReLU-7	[-1, 128, 256, 256]	0
Conv2d-8	[-1, 128, 256, 256]	147, 584
ReLU-9	[-1, 128, 256, 256]	0
MaxPool2d-10	[-1, 128, 128, 128]	0
Conv2d-11	[-1, 256, 128, 128]	295, 168
ReLU-12	[-1, 256, 128, 128]	0
Conv2d-13	[-1, 256, 128, 128]	590, 080
ReLU-14	[-1, 256, 128, 128]	0
Conv2d-15	[-1, 256, 128, 128]	590, 080
ReLU-16	[-1, 256, 128, 128]	0
MaxPool2d-17	[-1, 256, 64, 64]	0
Conv2d-18	[-1, 512, 64, 64]	1, 180, 160
ReLU-19	[-1, 512, 64, 64]	0
Conv2d-20	[-1, 512, 64, 64]	2, 359, 808
ReLU-21	[-1, 512, 64, 64]	0
Conv2d-22	[-1, 512, 64, 64]	2, 359, 808
ReLU-23	[-1, 512, 64, 64]	0
MaxPool2d-24	[-1, 512, 32, 32]	0
Conv2d-25	[-1, 512, 32, 32]	2, 359, 808
ReLU-26	[-1, 512, 32, 32]	0
Conv2d-27	[-1, 512, 32, 32]	2, 359, 808
ReLU-28	[-1, 512, 32, 32]	0
Conv2d-29	[-1, 512, 32, 32]	2, 359, 808
ReLU-30	[-1, 512, 32, 32]	0
MaxPool2d-31	[-1, 512, 16, 16]	0
ConvTranspose2d-32	[-1, 512, 32, 32]	2, 359, 808
ReLU-33	[-1, 512, 32, 32]	0
BatchNorm2d-34	[-1, 512, 32, 32]	1, 024
ConvTranspose2d-35	[-1, 256, 64, 64]	1, 179, 904
ReLU-36	[-1, 256, 64, 64]	0
BatchNorm2d-37	[-1, 256, 64, 64]	512
ConvTranspose2d-38	[-1, 128, 128, 128]	295, 040
ReLU-39	[-1, 128, 128, 128]	0
BatchNorm2d-40	[-1, 128, 128, 128]	256
ConvTranspose2d-41	[-1, 64, 256, 256]	73, 792
ReLU-42	[-1, 64, 256, 256]	0
BatchNorm2d-43	[-1, 64, 256, 256]	128
ConvTranspose2d-44	[-1, 32, 512, 512]	18, 464
ReLU-45	[-1, 32, 512, 512]	0
BatchNorm2d-46	[-1, 32, 512, 512]	64
Conv2d-47	[-1, 7, 512, 512]	231

3. Report mIoU of the improved model on the validation set.

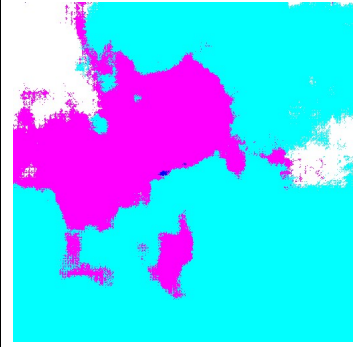
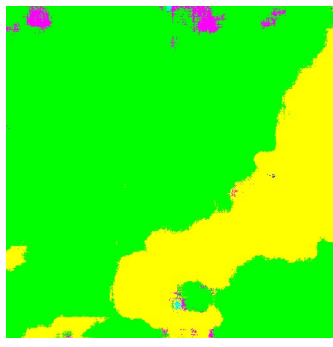
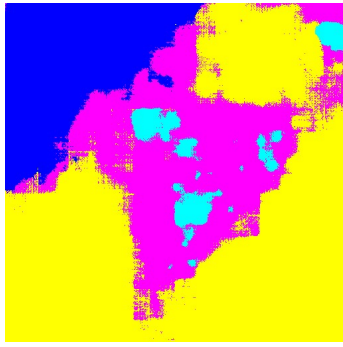
0.7151

4. Show the predicted segmentation mask of “validation/0010\_sat.jpg”, “validation/0097\_sat.jpg”, “validation/0107\_sat.jpg” during the early, middle, and the final stage during the training process of this improved model.

總共訓練 74 個 epoch

第幾個 epoch	predicted segmentation mask of “validation/0010_sat.jpg”	predicted segmentation mask of “validation/0097_sat.jpg”	predicted segmentation mask of “validation/0107_sat.jpg”
1			
31			
41			





### References

1. [https://github.com/kai860115/DLCV2020-FALL/tree/main/hw2/semantic\\_segmentation](https://github.com/kai860115/DLCV2020-FALL/tree/main/hw2/semantic_segmentation)
2. [https://github.com/overfitover/fcn\\_pytorch](https://github.com/overfitover/fcn_pytorch)