

Deep Learning for Computer Vision HW #1

姓名：蔡宇晴
學號：R08945050

Problem 1: Image classification (10%)

no collaborators

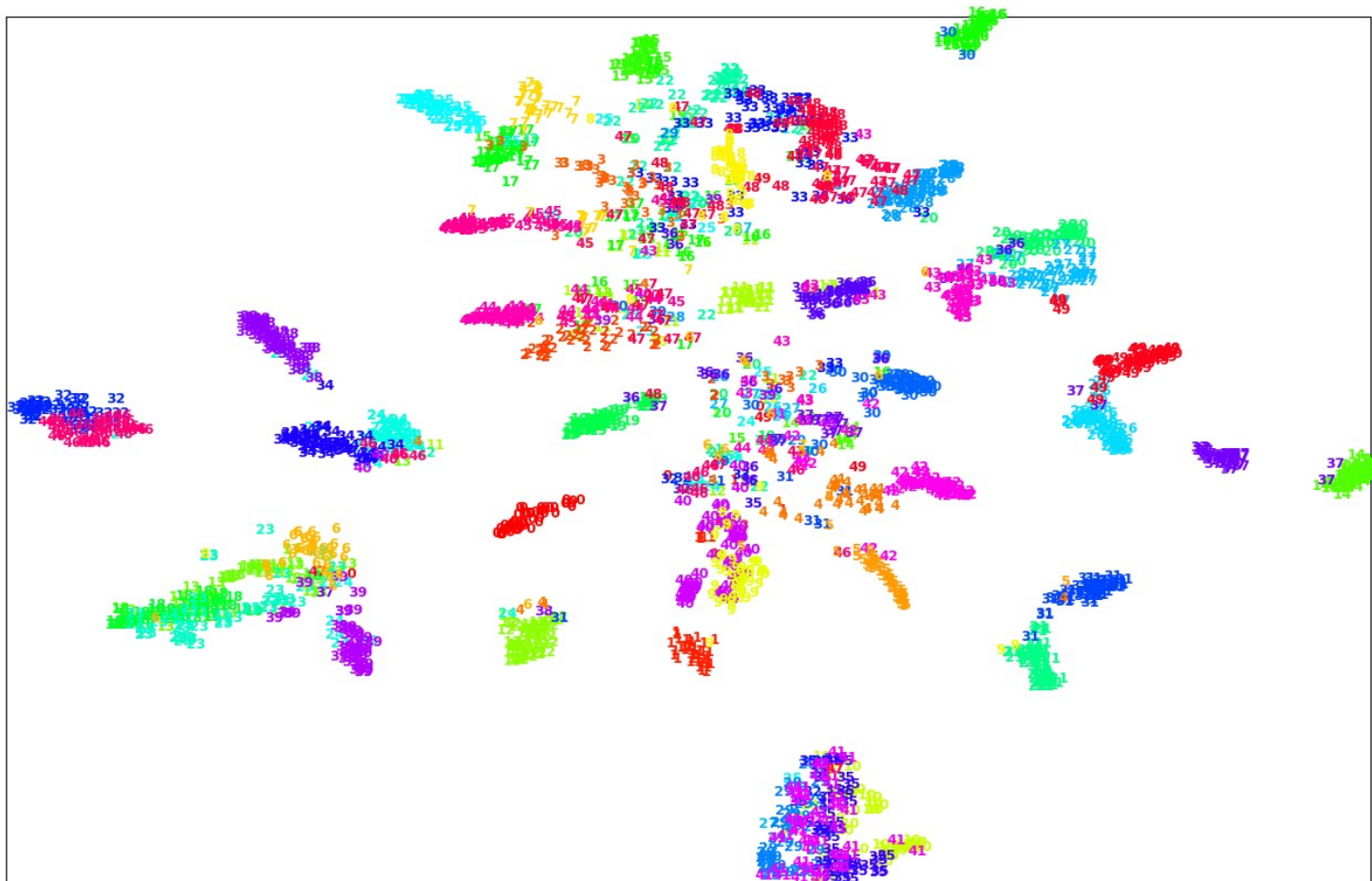
1. (2%) Print the network architecture of your model.

```
MyVGG16(  
  (features): Sequential(  
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (1): ReLU(inplace=True)  
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (3): ReLU(inplace=True)  
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (6): ReLU(inplace=True)  
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (8): ReLU(inplace=True)  
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (11): ReLU(inplace=True)  
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (13): ReLU(inplace=True)  
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (15): ReLU(inplace=True)  
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (18): ReLU(inplace=True)  
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (20): ReLU(inplace=True)  
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (22): ReLU(inplace=True)  
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (25): ReLU(inplace=True)  
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (27): ReLU(inplace=True)  
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (29): ReLU(inplace=True)  
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (classifier): Sequential(  
    (0): Dropout(p=0.5, inplace=False)  
    (1): Linear(in_features=8192, out_features=64, bias=True)  
    (2): ReLU(inplace=True)  
    (3): Dropout(p=0.5, inplace=False)  
    (4): Linear(in_features=64, out_features=50, bias=True)  
  )  
)
```

- (2%) Report accuracy of model on the validation set.

```
Accuracy train: 1.00  
Accuracy validate: 0.81
```

- (6%) Visualize the classification result on validation set by implementing t-SNE on output features of the second last layer. Briefly explain your result of tSNE visualization.



在倒數第2層的資料，相同屬性會在較近的距離，可以說明在靠近輸出層的資料已經有達到分類的效果

Problem 2: Semantic segmentation (30%)

1. (5%) Print the network architecture of your VGG16-FCN32s model.

```
MyFCN32(  
  (features): Sequential(  
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(100, 100))  
    (1): ReLU(inplace=True)  
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (3): ReLU(inplace=True)  
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)  
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (6): ReLU(inplace=True)  
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (8): ReLU(inplace=True)  
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)  
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (11): ReLU(inplace=True)  
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (13): ReLU(inplace=True)  
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (15): ReLU(inplace=True)  
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)  
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (18): ReLU(inplace=True)  
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (20): ReLU(inplace=True)  
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (22): ReLU(inplace=True)  
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)  
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (25): ReLU(inplace=True)  
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (27): ReLU(inplace=True)  
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
    (29): ReLU(inplace=True)  
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)  
  )  
  (score_fr): Sequential(  
    (0): Conv2d(512, 4096, kernel_size=(7, 7), stride=(1, 1))  
    (1): ReLU(inplace=True)  
    (2): Dropout(p=0.5, inplace=False)  
    (3): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))  
    (4): ReLU(inplace=True)  
    (5): Dropout(p=0.5, inplace=False)  
    (6): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))  
  )  
  (upscore): ConvTranspose2d(7, 7, kernel_size=(64, 64), stride=(32, 32), bias=False)  
)
```

2. (5%) Implement an improved model which performs better than your baseline model. Print the network architecture of this model.




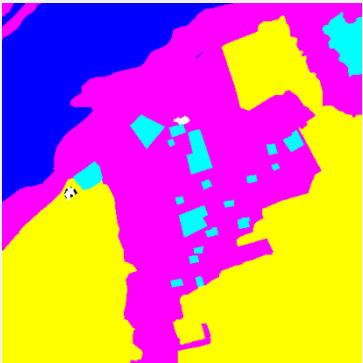
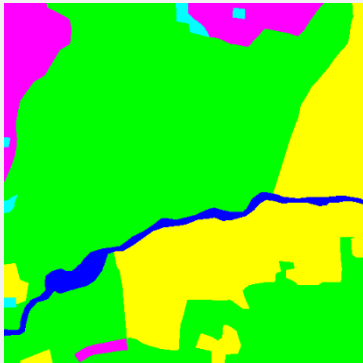
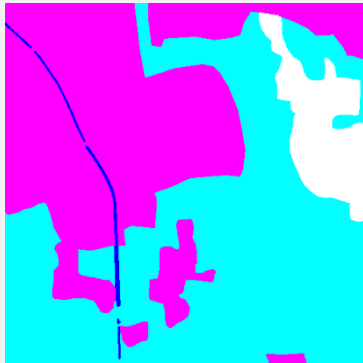



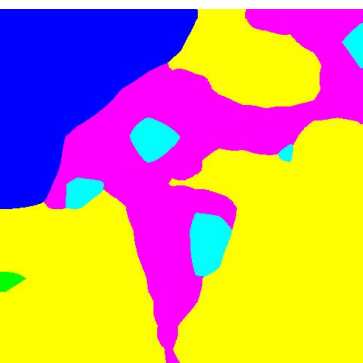


```
MyFCN8(
  (features3): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(100, 100))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
  )
  (features4): Sequential(
    (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
  )
  (features5): Sequential(
    (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
  )
  (score_pool3): Conv2d(256, 7, kernel_size=(1, 1), stride=(1, 1))
  (score_pool4): Conv2d(512, 7, kernel_size=(1, 1), stride=(1, 1))
  (score_fr): Sequential(
    (0): Conv2d(512, 4096, kernel_size=(7, 7), stride=(1, 1))
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
  )
  (upscore2): ConvTranspose2d(7, 7, kernel_size=(4, 4), stride=(2, 2), bias=False)
  (upscore_pool4): ConvTranspose2d(7, 7, kernel_size=(4, 4), stride=(2, 2), bias=False)
  (upscore8): ConvTranspose2d(7, 7, kernel_size=(16, 16), stride=(8, 8), bias=False)
)
```

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

1. FCN32s : 68.12%

2. FCN8s : 69.17%

4. (5%) 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.

	validation/0010_sat.jpg	validation/0097_sat.jpg	validation/0107_sat.jpg
Sat			
GT			
Epoch 1			
Epoch 20			

	validation/0010_sat.jpg	validation/0097_sat.jpg	validation/0107_sat.jpg
Epoch 40	