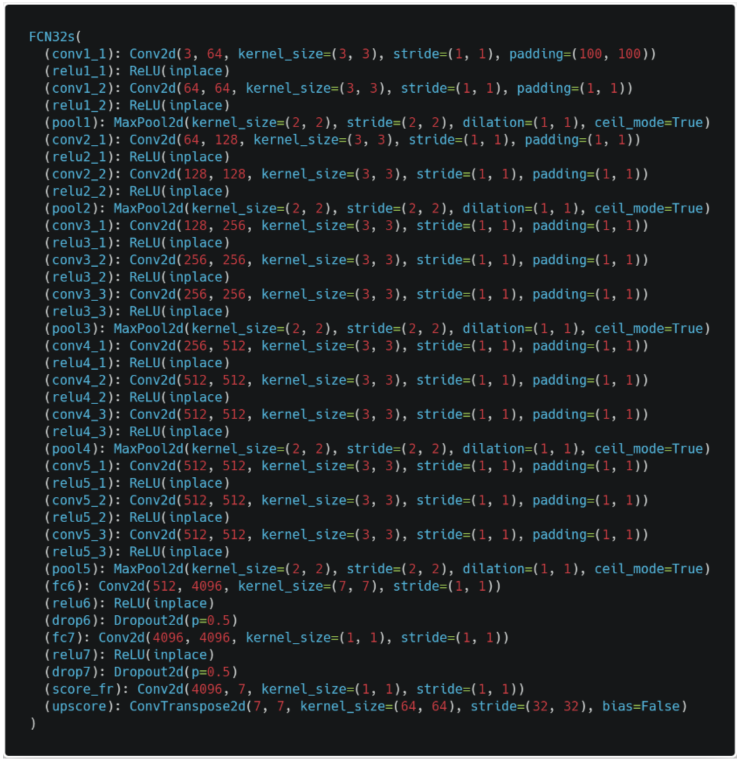
Name:汪智祥 Dep.:網媒所碩一 Student ID:R06944046

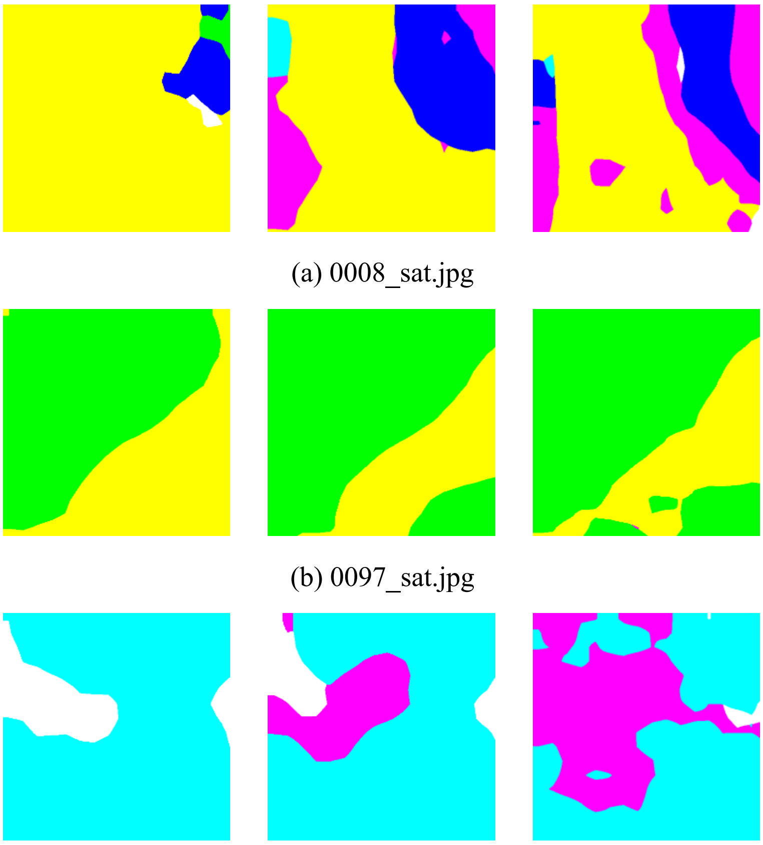
1. (5%) The network architecture of our VGG16-FCN32s model is shown in Fig. 1.



```
FCN32s(
  (conv1_1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(100, 100))
  (relu1_1): ReLU(inplace)
  (conv1_2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu1_2): ReLU(inplace)
  (pool1): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (conv2_1): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2_1): ReLU(inplace)
  (conv2_2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2_2): ReLU(inplace)
  (pool2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (conv3_1): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu3_1): ReLU(inplace)
  (conv3_2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu3_2): ReLU(inplace)
  (conv3_3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu3_3): ReLU(inplace)
  (pool3): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (conv4_1): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu4_1): ReLU(inplace)
  (conv4_2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu4_2): ReLU(inplace)
  (conv4_3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu4_3): ReLU(inplace)
  (pool4): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (conv5_1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu5_1): ReLU(inplace)
  (conv5_2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu5_2): ReLU(inplace)
  (conv5_3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu5_3): ReLU(inplace)
  (pool5): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (fc6): Conv2d(512, 4096, kernel_size=(7, 7), stride=(1, 1))
  (relu6): ReLU(inplace)
  (drop6): Dropout2d(p=0.5)
  (fc7): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
  (relu7): ReLU(inplace)
  (drop7): Dropout2d(p=0.5)
  (score_fr): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
  (upscore): ConvTranspose2d(7, 7, kernel_size=(64, 64), stride=(32, 32), bias=False)
)
```

Figure 1: Network architecture of the baseline model

2. (10%) The predicted segmentation mask of validation images {0008_sat.jpg, 0097_sat.jpg, 0107_sat.jpg} during the early, middle, and the final stage of the training stage are illustrate in Fig. 2.



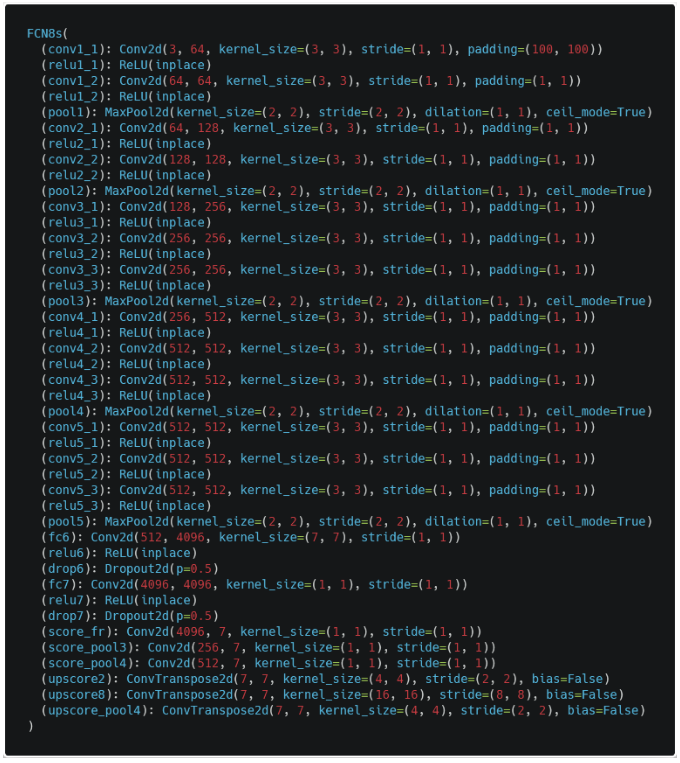
(a) 0008_sat.jpg

(b) 0097_sat.jpg

(c) 0107_sat.jpg

Figure 2: From left to right, epoch equals to 1, 5 and 80

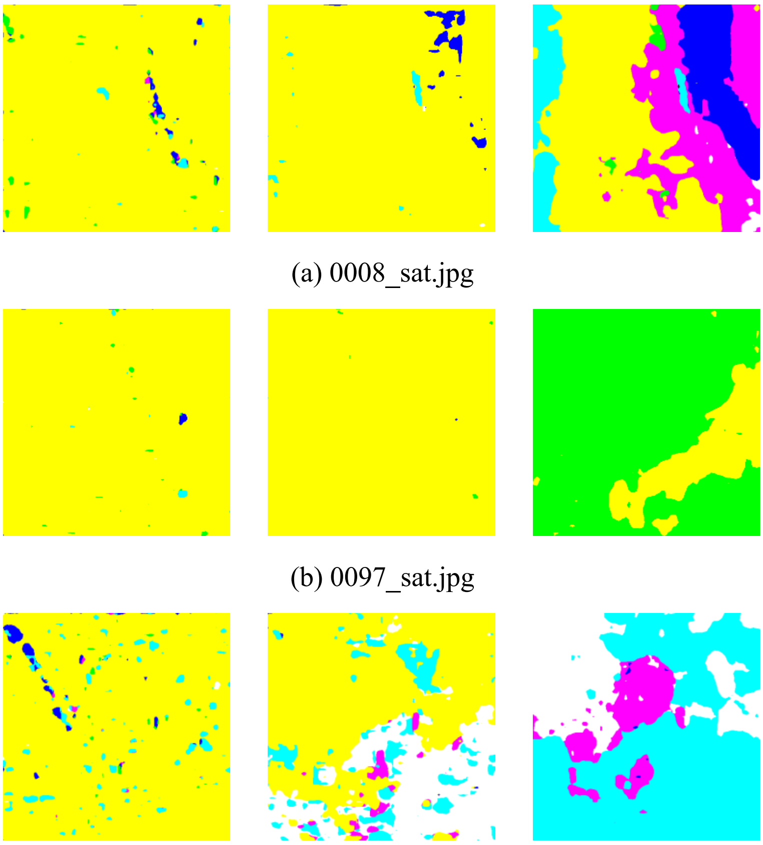
3. (15%) The network architecture of this improved model (VGG16-FCN8s model) is shown in Fig. 3.



```
FCN8s(
  (conv1_1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(100, 100))
  (relu1_1): ReLU(inplace)
  (conv1_2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu1_2): ReLU(inplace)
  (pool1): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (conv2_1): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2_1): ReLU(inplace)
  (conv2_2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2_2): ReLU(inplace)
  (pool2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (conv3_1): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu3_1): ReLU(inplace)
  (conv3_2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu3_2): ReLU(inplace)
  (conv3_3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu3_3): ReLU(inplace)
  (pool3): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (conv4_1): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu4_1): ReLU(inplace)
  (conv4_2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu4_2): ReLU(inplace)
  (conv4_3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu4_3): ReLU(inplace)
  (pool4): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (conv5_1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu5_1): ReLU(inplace)
  (conv5_2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu5_2): ReLU(inplace)
  (conv5_3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu5_3): ReLU(inplace)
  (pool5): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=True)
  (fc6): Conv2d(512, 4096, kernel_size=(7, 7), stride=(1, 1))
  (relu6): ReLU(inplace)
  (drop6): Dropout2d(p=0.5)
  (fc7): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
  (relu7): ReLU(inplace)
  (drop7): Dropout2d(p=0.5)
  (score_fr): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
  (score_pool3): Conv2d(256, 7, kernel_size=(1, 1), stride=(1, 1))
  (score_pool4): Conv2d(512, 7, kernel_size=(1, 1), stride=(1, 1))
  (upscore2): 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)
  (upscore_pool4): ConvTranspose2d(7, 7, kernel_size=(4, 4), stride=(2, 2), bias=False)
)
```

Figure 3: Network architecture of the improved model

4. (10%) The predicted segmentation mask of validation images {0008_sat.jpg, 0097_sat.jpg, 0107_sat.jpg} during the early, middle, and the final stage during training the improved model is shown in Fig. 4. It is trained use the pre-trained VGG16 model.



(a) 0008_sat.jpg

(b) 0097_sat.jpg

(c) 0107_sat.jpg

Figure 4: From left to right, epoch equals to 1, 20 and 40

5. (15%) mIoU score of both models on the validation set: *a*) baseline model: 69.8, *b*) improved model 71.0. The improved model combine the coarse and fine prediction together, much more detail information makes it works better than the FCN32s.