Homework #2

November 10, 2020

Problem 1: Image Classification

1. Print the network architecture

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 224, 224]           1,792
       BatchNorm2d-2         [-1, 64, 224, 224]             128
              ReLU-3         [-1, 64, 224, 224]               0
            Conv2d-4         [-1, 64, 224, 224]          36,928
       BatchNorm2d-5         [-1, 64, 224, 224]             128
              ReLU-6         [-1, 64, 224, 224]               0
         MaxPool2d-7         [-1, 64, 112, 112]               0
            Conv2d-8        [-1, 128, 112, 112]          73,856
       BatchNorm2d-9        [-1, 128, 112, 112]             256
             ReLU-10        [-1, 128, 112, 112]               0
           Conv2d-11        [-1, 128, 112, 112]         147,584
      BatchNorm2d-12        [-1, 128, 112, 112]             256
             ReLU-13        [-1, 128, 112, 112]               0
        MaxPool2d-14          [-1, 128, 56, 56]               0
           Conv2d-15          [-1, 256, 56, 56]         295,168
      BatchNorm2d-16          [-1, 256, 56, 56]             512
             ReLU-17          [-1, 256, 56, 56]               0
           Conv2d-18          [-1, 256, 56, 56]         590,080
      BatchNorm2d-19          [-1, 256, 56, 56]             512
             ReLU-20          [-1, 256, 56, 56]               0
           Conv2d-21          [-1, 256, 56, 56]         590,080
      BatchNorm2d-22          [-1, 256, 56, 56]             512
             ReLU-23          [-1, 256, 56, 56]               0
        MaxPool2d-24          [-1, 256, 28, 28]               0
           Conv2d-25          [-1, 512, 28, 28]       1,180,160
      BatchNorm2d-26          [-1, 512, 28, 28]           1,024
             ReLU-27          [-1, 512, 28, 28]               0
           Conv2d-28          [-1, 512, 28, 28]       2,359,808
      BatchNorm2d-29          [-1, 512, 28, 28]           1,024
             ReLU-30          [-1, 512, 28, 28]               0
           Conv2d-31          [-1, 512, 28, 28]       2,359,808
      BatchNorm2d-32          [-1, 512, 28, 28]           1,024
             ReLU-33          [-1, 512, 28, 28]               0
        MaxPool2d-34          [-1, 512, 14, 14]               0
           Conv2d-35          [-1, 512, 14, 14]       2,359,808
      BatchNorm2d-36          [-1, 512, 14, 14]           1,024
             ReLU-37          [-1, 512, 14, 14]               0
           Conv2d-38          [-1, 512, 14, 14]       2,359,808
      BatchNorm2d-39          [-1, 512, 14, 14]           1,024
             ReLU-40          [-1, 512, 14, 14]               0
           Conv2d-41          [-1, 512, 14, 14]       2,359,808
```

```
      BatchNorm2d-42          [-1, 512, 14, 14]           1,024
             ReLU-43          [-1, 512, 14, 14]               0
        MaxPool2d-44            [-1, 512, 7, 7]               0
AdaptiveAvgPool2d-45            [-1, 512, 7, 7]               0
           Linear-46                  [-1, 4096]     102,764,544
             ReLU-47                  [-1, 4096]               0
          Dropout-48                  [-1, 4096]               0
           Linear-49                  [-1, 4096]      16,781,312
           Linear-50                  [-1, 4096]     102,764,544
             ReLU-51                  [-1, 4096]               0
          Dropout-52                  [-1, 4096]               0
           Linear-53                  [-1, 4096]      16,781,312
             ReLU-54                  [-1, 4096]               0
          Dropout-55                  [-1, 4096]               0
           Linear-56                    [-1, 50]         204,850
================================================================
Total params: 254,019,698
Trainable params: 254,019,698
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 322.26
Params size (MB): 969.01
Estimated Total Size (MB): 1291.84
----------------------------------------------------------------
```

2. Report accuracy of model on the validation set.

```python
correct = 0
with open('../test_pred.csv', 'r') as f:
    f.readline()
    lines = f.readlines()
    for line in lines:
        if line.split('_')[0] == line.split(',')[-1].split('\n')[0]:
            correct += 1

print('Accuracy: %f' % (correct/len(lines)))
```

```
Accuracy: 0.846400
```

Figure 1: for 1.2

3. Visualize the classification result on validation set by implementing t-SNE on output features of the second last layer.

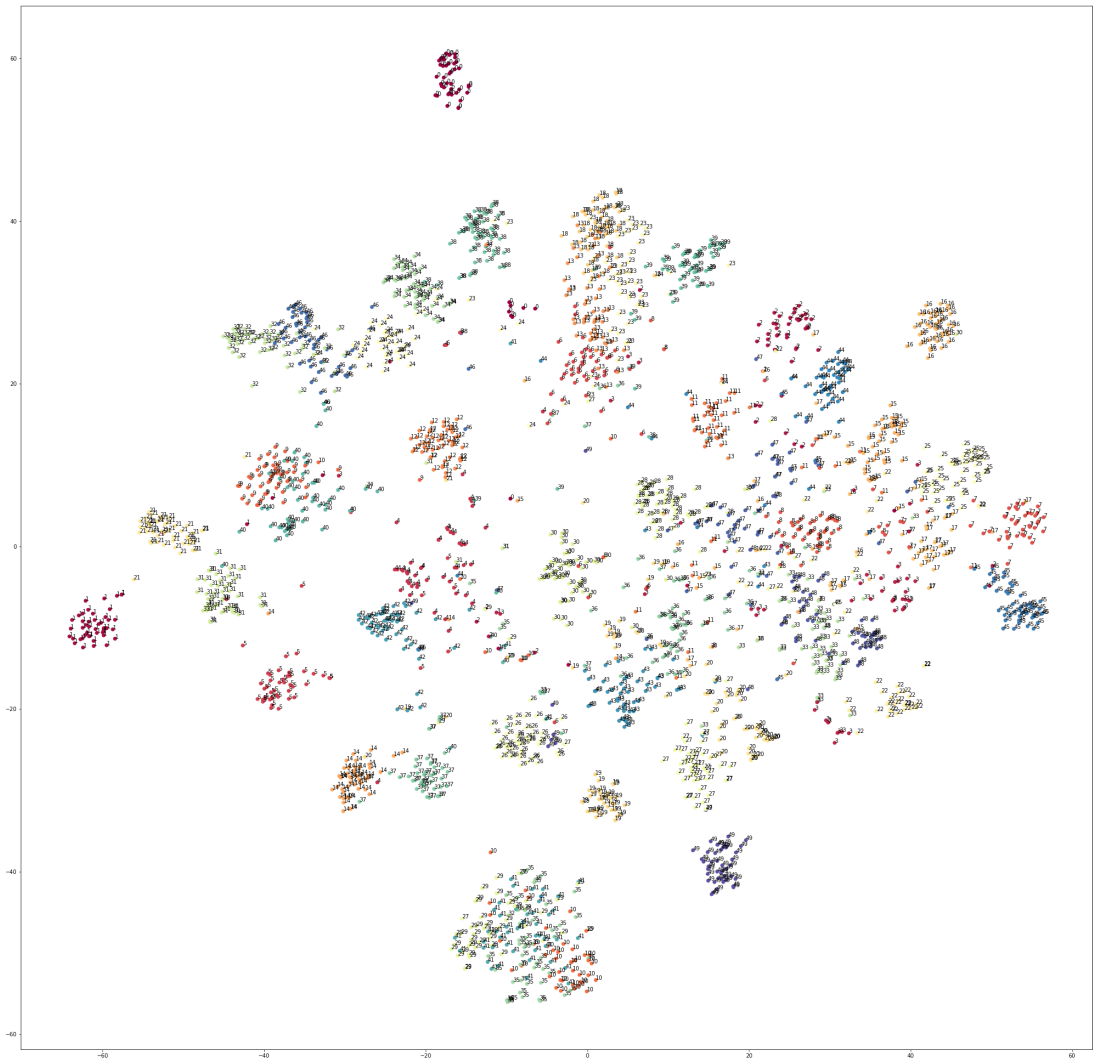
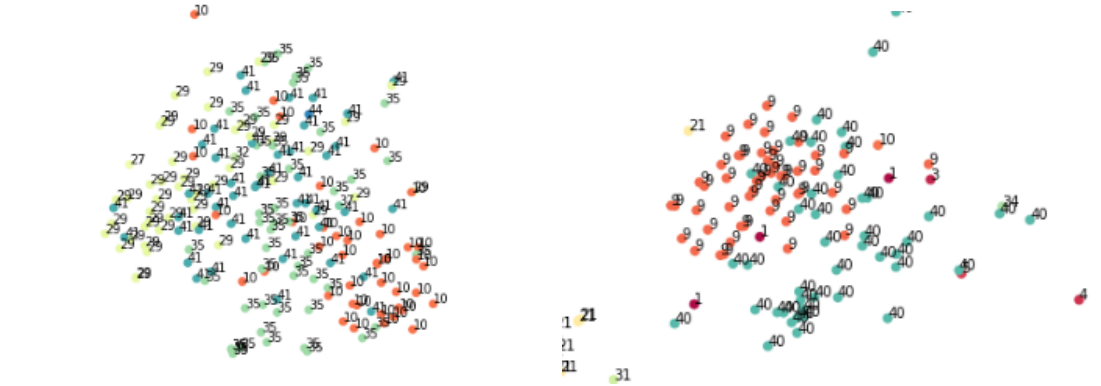


Figure 2: for 1.3

可以看出來同一種類別投影在 2d 平面會集中在一群，將其放大來看如下圖，可以看到 (22, 33, 47, 48)、(13, 18, 23)、(10, 29, 35, 41)、(9, 40) 這幾個組合會擠一塊，可能在分類上這些組合的成果容易辨別錯誤，從 confusion matrix (Figure: 3) 看可以證實這個推論，這幾個點確實容易辨別為彼此。

(a) 22, 33, 47, 48 (b) 13, 18, 23



(c) 10, 29, 35, 41 (d) 9, 40

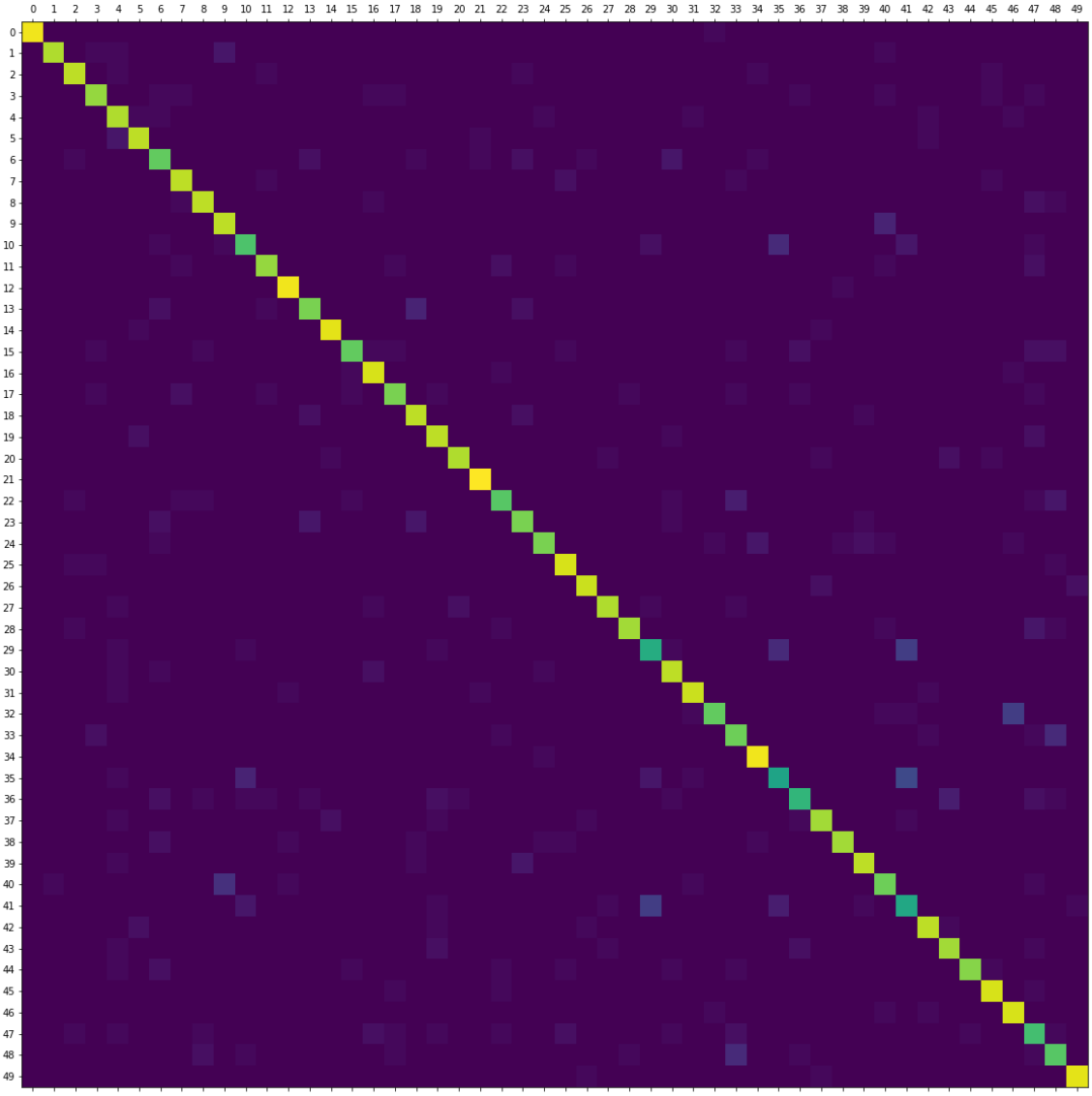


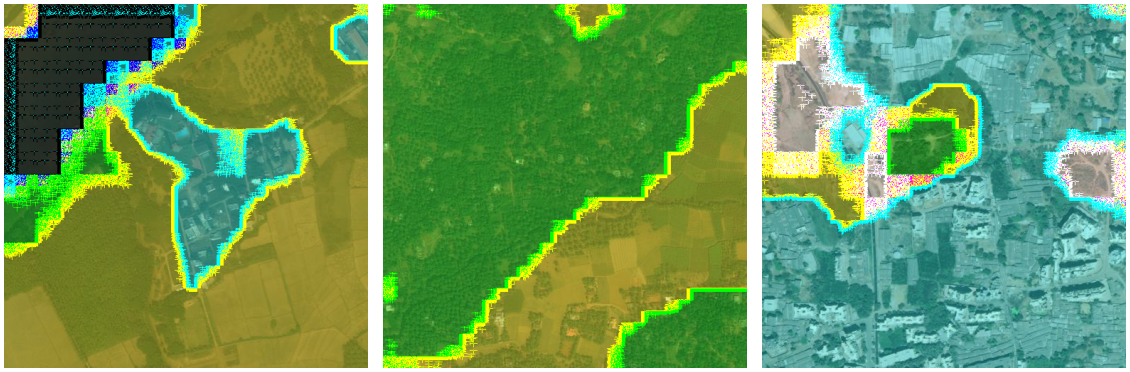
Figure 3: confusion matrix (x 軸為 predict, y 軸為 true)

Problem 2: Semantic segmentation

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

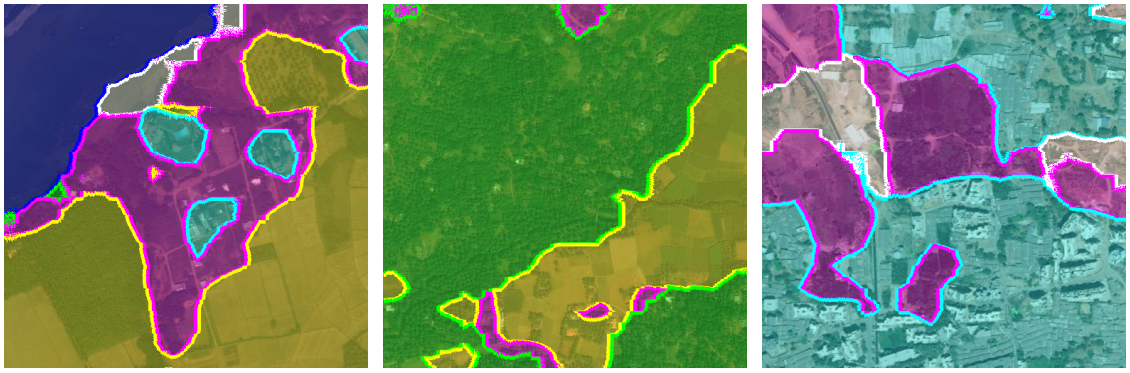
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 512, 512]           1,792
              ReLU-2         [-1, 64, 512, 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
           Conv2d-32         [-1, 4096, 16, 16]       2,101,248
             ReLU-33         [-1, 4096, 16, 16]               0
        Dropout2d-34         [-1, 4096, 16, 16]               0
           Conv2d-35         [-1, 4096, 16, 16]      16,781,312
             ReLU-36         [-1, 4096, 16, 16]               0
        Dropout2d-37         [-1, 4096, 16, 16]               0
           Conv2d-38            [-1, 7, 16, 16]          28,679
  ConvTranspose2d-39          [-1, 7, 544, 544]         200,704
================================================================
Total params: 33,826,631
Trainable params: 33,826,631
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 3.00
Forward/backward pass size (MB): 1204.82
Params size (MB): 129.04
Estimated Total Size (MB): 1336.86
----------------------------------------------------------------
```

2. 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 stage.



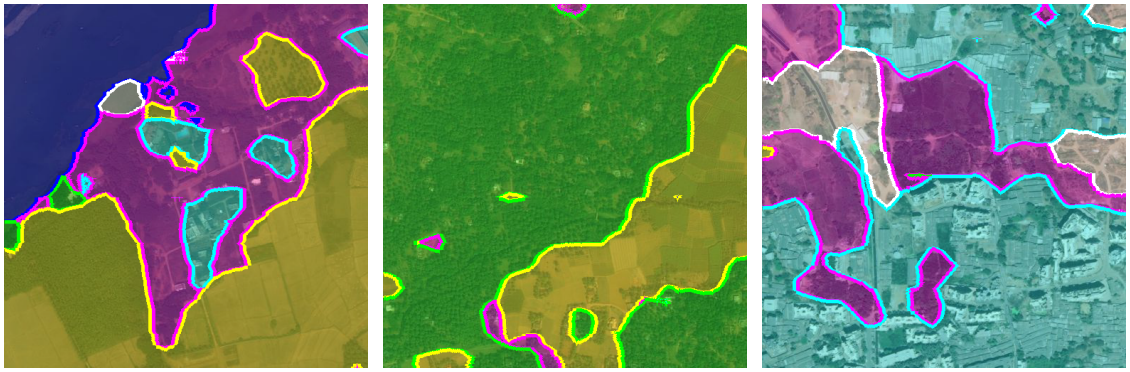
(a) 0010 (b) 0097 (c) 0107

Figure 4: After 2nd epoch



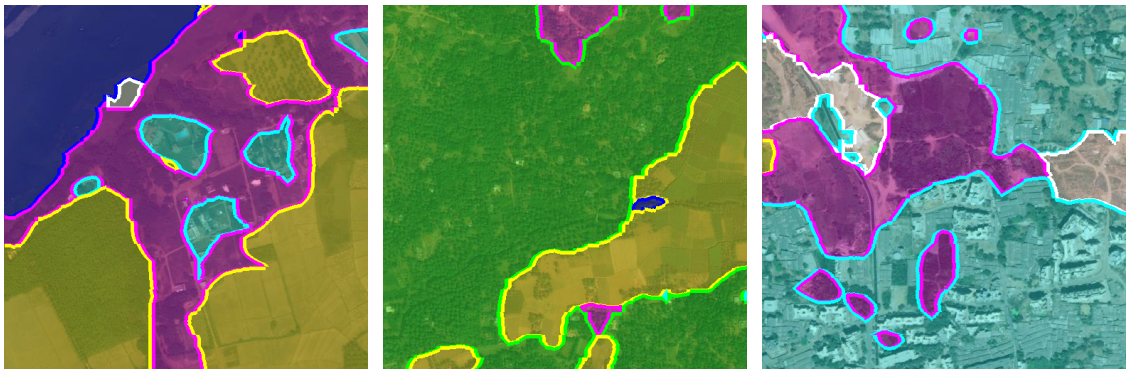
(a) 0010 (b) 0097 (c) 0107

Figure 5: After 18th epoch



(a) 0010 (b) 0097 (c) 0107

Figure 6: After 36th epoch



(a) 0010 (b) 0097 (c) 0107

Figure 7: After 83rd epoch (best mean iou)

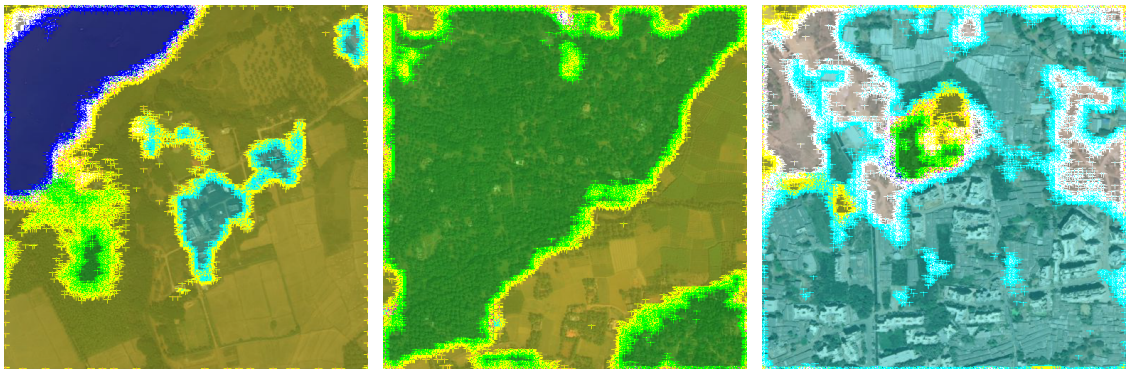
3. Implement an improved model which performs better than your baseline model. Print the network architecture of this model.

VGG16-FCN8s 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
Conv2d-32	[-1, 4096, 16, 16]	2,101,248
ReLU-33	[-1, 4096, 16, 16]	0
Dropout2d-34	[-1, 4096, 16, 16]	0
Conv2d-35	[-1, 4096, 16, 16]	16,781,312

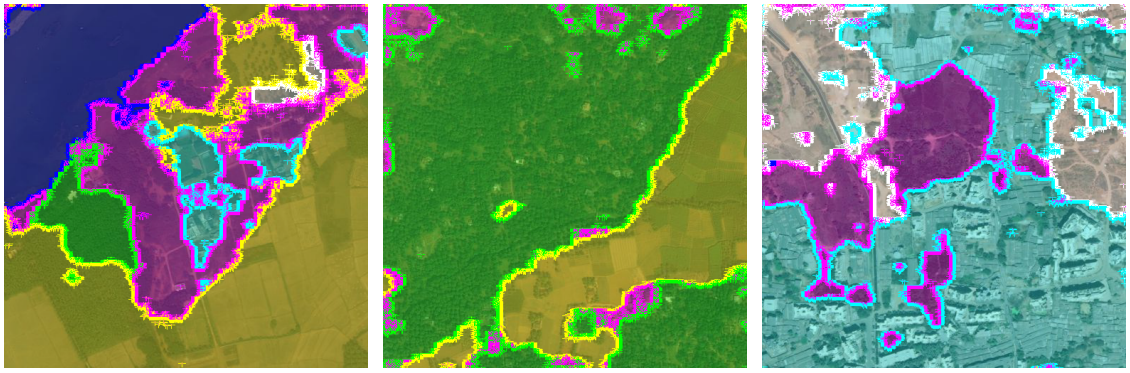
```
              ReLU-36          [-1, 4096, 16, 16]               0
           Dropout2d-37          [-1, 4096, 16, 16]               0
              Conv2d-38             [-1, 7, 16, 16]          28,679
    ConvTranspose2d-39             [-1, 7, 34, 34]             784
              Conv2d-40             [-1, 7, 32, 32]           3,591
    ConvTranspose2d-41             [-1, 7, 66, 66]             784
              Conv2d-42             [-1, 7, 64, 64]           1,799
    ConvTranspose2d-43           [-1, 7, 520, 520]          12,544
================================================================
Total params: 33,645,429
Trainable params: 33,645,429
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 3.00
Forward/backward pass size (MB): 1204.02
Params size (MB): 128.35
Estimated Total Size (MB): 1335.37
----------------------------------------------------------------
```

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.



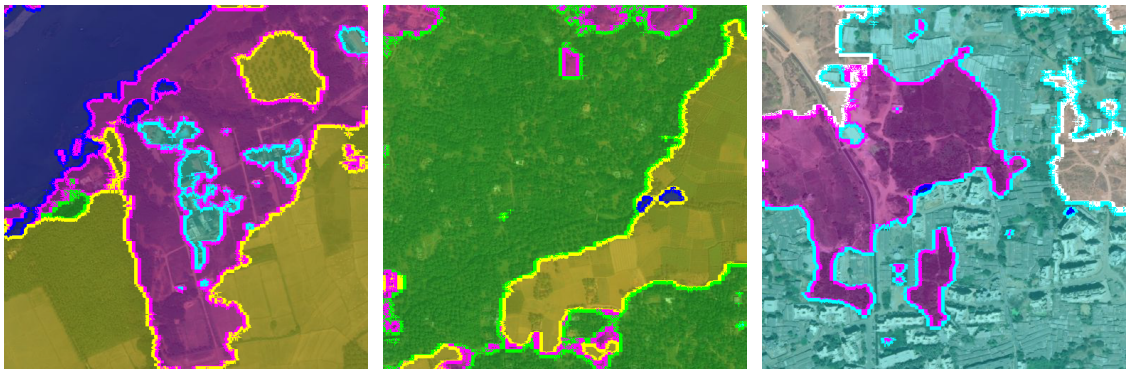
(a) 0010 (b) 0097 (c) 0107

Figure 8: After 2nd epoch



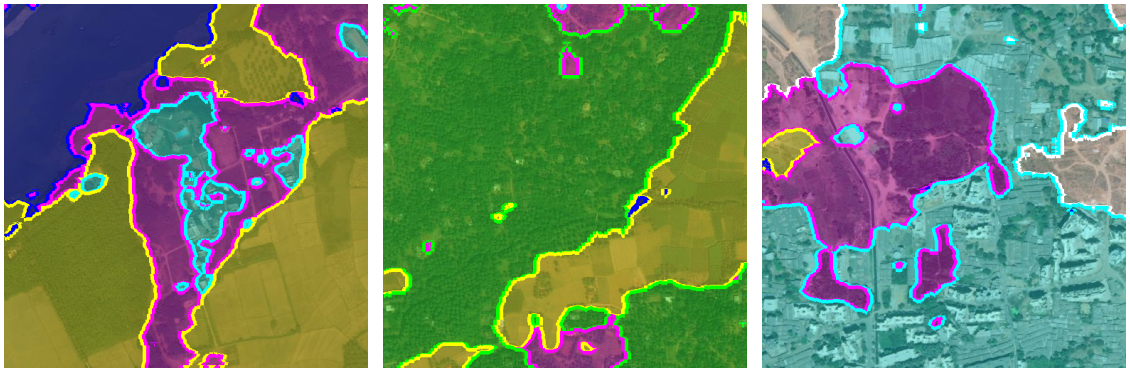
(a) 0010 (b) 0097 (c) 0107

Figure 9: After 18th epoch



(a) 0010 (b) 0097 (c) 0107

Figure 10: After 38th epoch



(a) 0010 (b) 0097 (c) 0107

Figure 11: After 92th epoch (best mean iou)

5. Report mIoU score of both models on the validation set.

VGG16-FCN32s model

```
class #0 : 0.76421
class #1 : 0.89086
class #2 : 0.38150
class #3 : 0.80907
class #4 : 0.77154
class #5 : 0.68555

mean_iou: 0.717121
```

VGG16-FCN8s (improved) model

```
class #0 : 0.77172
class #1 : 0.89884
class #2 : 0.41808
class #3 : 0.81318
class #4 : 0.78343
class #5 : 0.71257

mean_iou: 0.732969
```

FCN32 可能因為只用 CNN 的最後一層，所以有些比較細節的部份會比起 FCN8 來的差，可以由第 2、4 小題看出，此外為證明多加 CNN 上層去做 upsample 會比較好，這裡也做了 FCN16，其 validation dataset 的 mean iou 最高為 0.732096，可以看出多加 pool4，影響很大，pool3 的影響反而沒有那麼大；此外 data augmentation (random flip) 的幫助也很大，FCN32 的 mean iou 從 0.703049 進步到 0.717121，而 FCN8 從 0.713941 到 0.732969。

Reference

1. data augmentation for semantic segmentation (https://discuss.pytorch.org/t/torchvision-transfors-how-to-perform-identical-transform-on-both-image-and-target/10606/31?fbclid=IwAR1zHP1XKmi_y5fS3ARh2Mtl0MUGrjI4KMhPQJWjiiX4-zer3Foav26PP2A)
2. VGG16-FCN (https://github.com/affromero/FCN)