

DLCV Hw3 Report

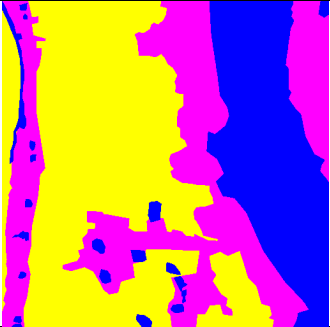
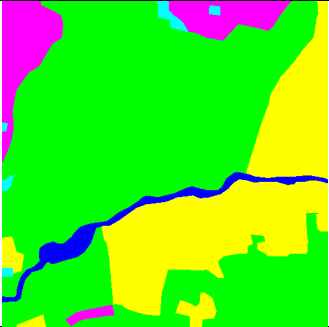
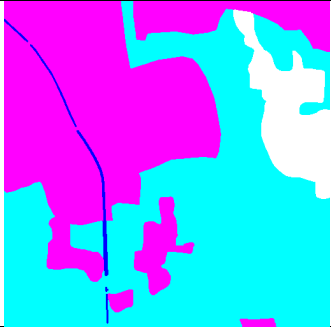
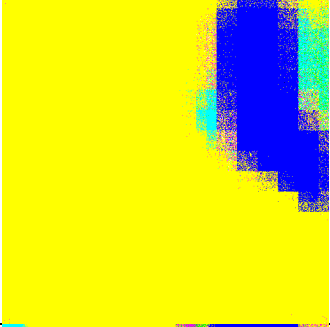
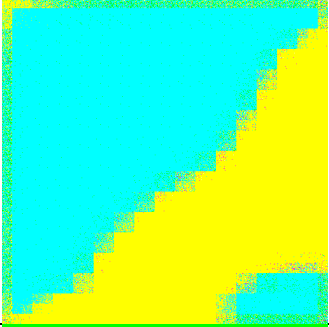
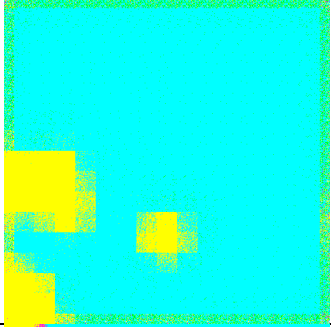

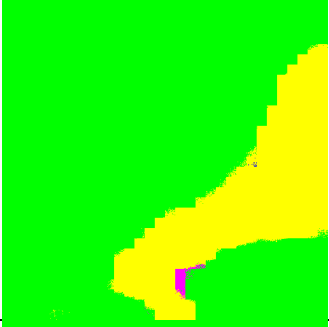


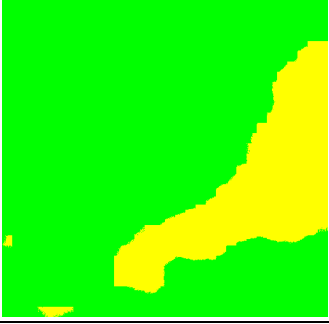

Name: 蔡昕宇 Dep.:電機三 Student ID:B04901096

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

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 512, 512, 3)	0
block1_conv1 (Conv2D)	(None, 512, 512, 64)	1792
block1_conv2 (Conv2D)	(None, 512, 512, 64)	36928
block1_pool (MaxPooling2D)	(None, 256, 256, 64)	0
block2_conv1 (Conv2D)	(None, 256, 256, 128)	73856
block2_conv2 (Conv2D)	(None, 256, 256, 128)	147584
block2_pool (MaxPooling2D)	(None, 128, 128, 128)	0
block3_conv1 (Conv2D)	(None, 128, 128, 256)	295168
block3_conv2 (Conv2D)	(None, 128, 128, 256)	590080
block3_conv3 (Conv2D)	(None, 128, 128, 256)	590080
block3_pool (MaxPooling2D)	(None, 64, 64, 256)	0
block4_conv1 (Conv2D)	(None, 64, 64, 512)	1180160
block4_conv2 (Conv2D)	(None, 64, 64, 512)	2359808
block4_conv3 (Conv2D)	(None, 64, 64, 512)	2359808
block4_pool (MaxPooling2D)	(None, 32, 32, 512)	0
block5_conv1 (Conv2D)	(None, 32, 32, 512)	2359808
block5_conv2 (Conv2D)	(None, 32, 32, 512)	2359808
block5_conv3 (Conv2D)	(None, 32, 32, 512)	2359808
block5_pool (MaxPooling2D)	(None, 16, 16, 512)	0
block6_conv1 (Conv2D)	(None, 16, 16, 4096)	18878464
dropout_1 (Dropout)	(None, 16, 16, 4096)	0
block6_conv2 (Conv2D)	(None, 16, 16, 4096)	16781312
dropout_2 (Dropout)	(None, 16, 16, 4096)	0
score (Conv2D)	(None, 16, 16, 7)	28679
block6_upsample (Conv2DTrans)	(None, 512, 512, 7)	200704
Total params: 50,603,847		
Trainable params: 50,603,847		
Non-trainable params: 0		
Training on VGG16-FCN32s.....		

2. (10%) Show the predicted segmentation mask of validation/0008_sat.jpg, validation/0097_sat.jpg, validation/0107_sat.jpg during the early, middle, and the final stage during the training stage. (For example, results of 1st, 10th, 20th epoch)

Results of VGG16-FCN32s during training.

epoch	validation/0008_sat.jpg	validation/0097_sat.jpg	validation/0107_sat.jpg
target			
1			
10			
20			

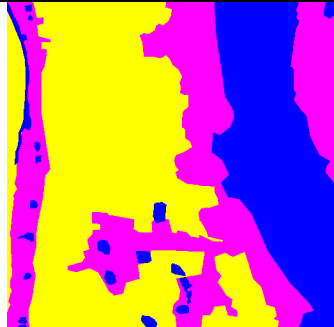
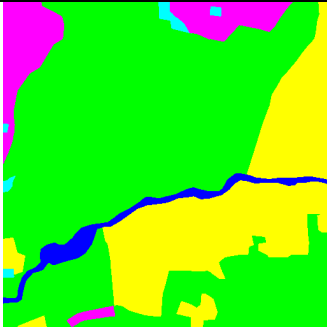
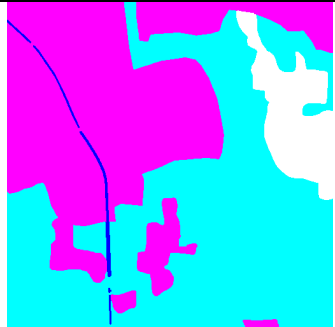

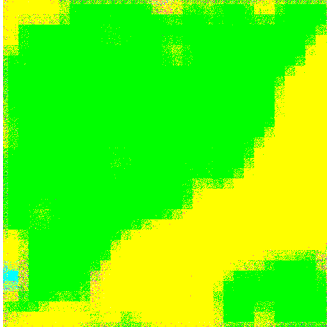
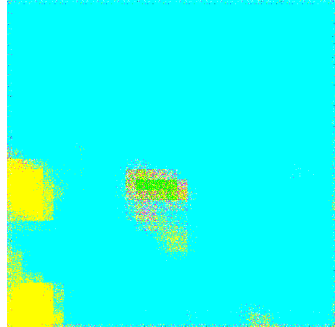
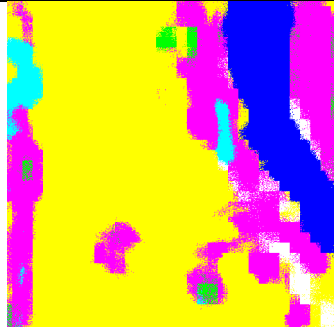
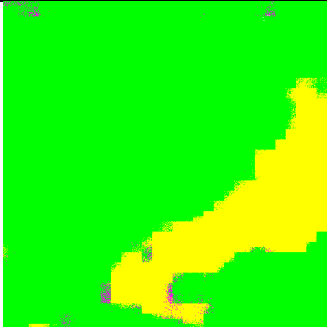
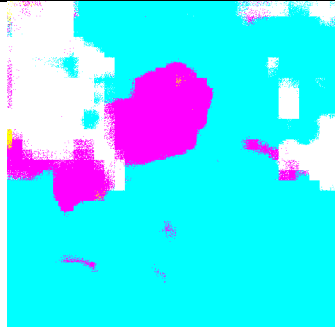

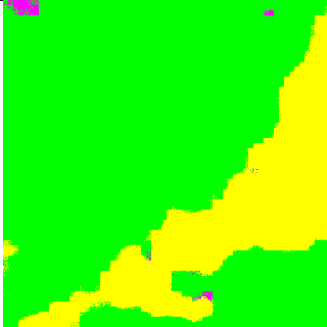

- (15%) Implement an improved model which performs better than your baseline model. Print the network architecture of this model.

Implement VGG16-FCN16s for improve model

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 512, 512, 3)	0	
block1_conv1 (Conv2D)	(None, 512, 512, 64)	1792	input_1[0][0]
block1_conv2 (Conv2D)	(None, 512, 512, 64)	36928	block1_conv1[0][0]
block1_pool (MaxPooling2D)	(None, 256, 256, 64)	0	block1_conv2[0][0]
block2_conv1 (Conv2D)	(None, 256, 256, 128)	73856	block1_pool[0][0]
block2_conv2 (Conv2D)	(None, 256, 256, 128)	147584	block2_conv1[0][0]
block2_pool (MaxPooling2D)	(None, 128, 128, 128)	0	block2_conv2[0][0]
block3_conv1 (Conv2D)	(None, 128, 128, 256)	295168	block2_pool[0][0]
block3_conv2 (Conv2D)	(None, 128, 128, 256)	590080	block3_conv1[0][0]
block3_conv3 (Conv2D)	(None, 128, 128, 256)	590080	block3_conv2[0][0]
block3_pool (MaxPooling2D)	(None, 64, 64, 256)	0	block3_conv3[0][0]
block4_conv1 (Conv2D)	(None, 64, 64, 512)	1180160	block3_pool[0][0]
block4_conv2 (Conv2D)	(None, 64, 64, 512)	2359808	block4_conv1[0][0]
block4_conv3 (Conv2D)	(None, 64, 64, 512)	2359808	block4_conv2[0][0]
block4_pool (MaxPooling2D)	(None, 32, 32, 512)	0	block4_conv3[0][0]
block5_conv1 (Conv2D)	(None, 32, 32, 512)	2359808	block4_pool[0][0]
block5_conv2 (Conv2D)	(None, 32, 32, 512)	2359808	block5_conv1[0][0]
block5_conv3 (Conv2D)	(None, 32, 32, 512)	2359808	block5_conv2[0][0]
block5_pool (MaxPooling2D)	(None, 16, 16, 512)	0	block5_conv3[0][0]
block6_conv1 (Conv2D)	(None, 16, 16, 4096)	18878464	block5_pool[0][0]
dropout_1 (Dropout)	(None, 16, 16, 4096)	0	block6_conv1[0][0]
block6_conv2 (Conv2D)	(None, 16, 16, 4096)	16781312	dropout_1[0][0]
dropout_2 (Dropout)	(None, 16, 16, 4096)	0	block6_conv2[0][0]
score (Conv2D)	(None, 16, 16, 7)	28679	dropout_2[0][0]
block6_upsample (Conv2DTranspos	(None, 34, 34, 7)	791	score[0][0]
cropping2d_1 (Cropping2D)	(None, 32, 32, 7)	0	block6_upsample[0][0]
score_pool4 (Conv2D)	(None, 32, 32, 7)	3591	block4_pool[0][0]
add_1 (Add)	(None, 32, 32, 7)	0	cropping2d_1[0][0] score_pool4[0][0]
block7_upsample (Conv2DTranspos	(None, 512, 512, 7)	50176	add_1[0][0]
Total params: 50,457,701			
Trainable params: 50,457,701			
Non-trainable params: 0			
Training on VGG16-FCN16s.....			

4. (10%) Show the predicted segmentation mask of validation/0008_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.

Results of VGG16-FCN16s during training.

epoch	validation/0008_sat.jpg	validation/0097_sat.jpg	validation/0107_sat.jpg
target			
1			
10			
20			

5. (15%) Report mIoU score of both models on the validation set. Discuss the reason why the improved model performs better than the baseline one. You may conduct some experiments and show some evidences to support your discussion.

Calculate mean IoU at epoch 20

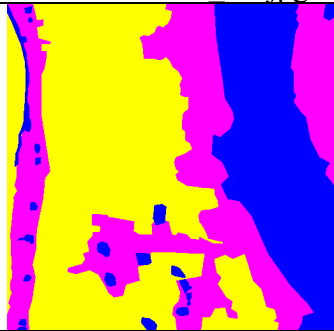
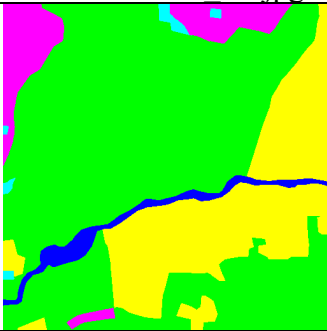
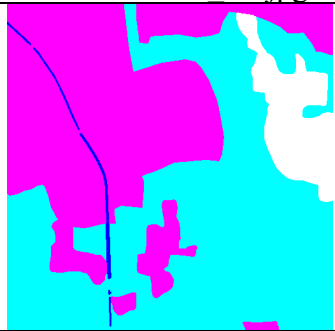
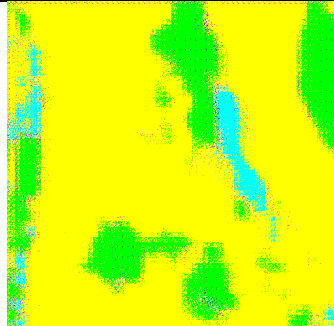
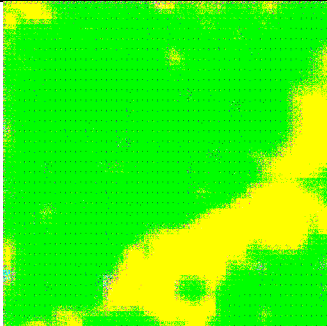
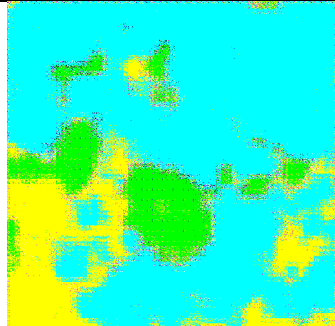
acc/mIoU	VGG16-FCN32s	VGG16-FCN16s
Class #0	0.74253	0.74944
Class #1	0.86813	0.87892
Class #2	0.30609	0.28751
Class #3	0.77914	0.77799
Class #4	0.72905	0.72519
Class #5	0.61108	0.65657
Mean IOU score	0.672671	0.679271

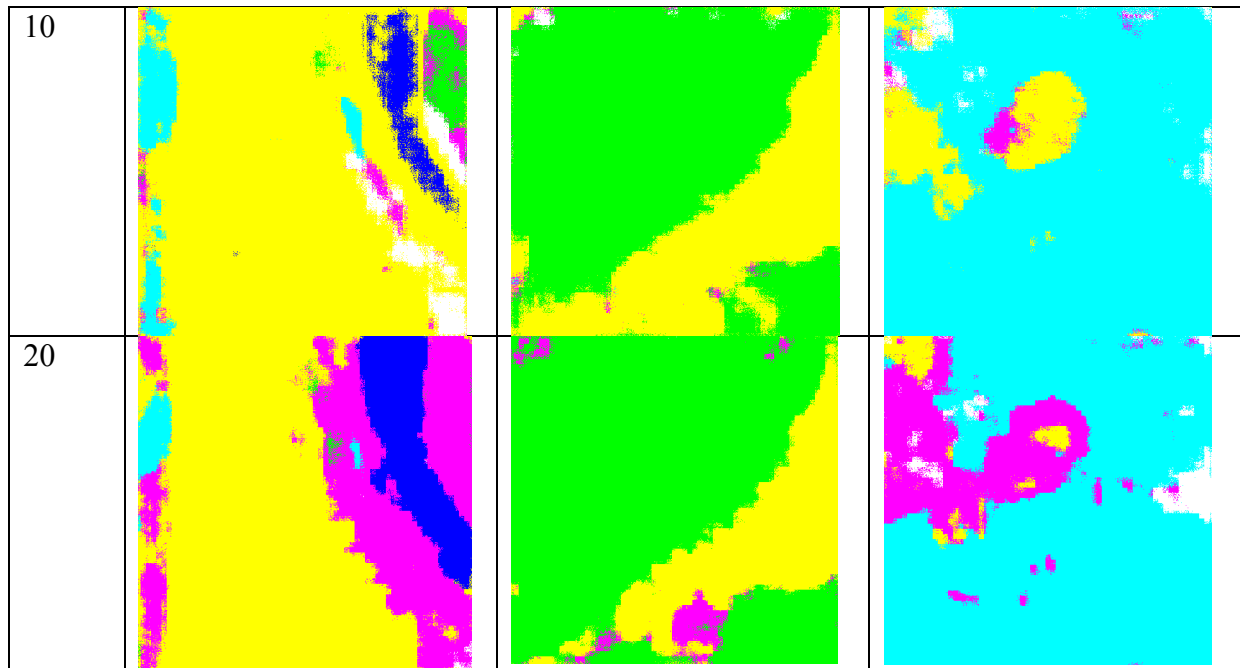
Observations:

FCN16s is a slightly improvement model from FCN32s. From the chart above, there is no obvious improvement in mean IOU between two models, but we see that in these two models, the predictions of each classes are slightly different. For example, class #2 in FCN32s is slightly higher than that in FCN16s, where Class #5 is much higher in FCN16s than that in FCN32s. Others seems no obvious difference between two model.

Mainly reason is that FCN16s outputs two layer from encoding, i.e. the data after the maxpooling layer of block 4 and the output layer of encoding. It seems that the model learned some information from the encoding layer, and more information from encoding in FCN16s.

In the experiment implement VGG16-FCN8s for experiment, I generate same figures in the questions above, in order to see if it satisfies our assumption just mentioned.

epoch	validation/0008 sat.jpg	validation/0097 sat.jpg	validation/0107 sat.jpg
target			
1			



We observe that FCN8s, the result seems not too good. It may result in the model did not initialize at a good spot, thus we can observe a large difference in prediction in 0008_mask.png and target in epoch 1, but it still can fit the target gradually afterwards. Mean IoU at epoch 20 is shown in the following chart.

acc/mIoU	VGG16-FCN8s
Class #0	0.76292
Class #1	0.87474
Class #2	0.26423
Class #3	0.73023
Class #4	0.71806
Class #5	0.61627
Mean IOU score	0.661078

The mean IoU result is not higher than that of other models. It may need more epochs to train the model. However, we can see that FCN8s can generate more small pieces in prediction, because it may have more information from encoding, just like we argued above.

6. (5%) [bonus] Calculate the result of $d/dw G(w)$:

objective function:

$$G(w) = - \sum_n [t^{(n)} \log x(z^{(n)}; w) + (1 - t^{(n)}) \log (1 - x(z^{(n)}; w))] \geq 0$$

$$w^* = \arg \min_w G(w) \quad \text{choose the weights that minimise the network's surprise about the training data}$$

$$\frac{d}{dw} G(w) = \sum_n \frac{dG(w)}{dx^{(n)}} \frac{dx^{(n)}}{dw} = - \sum_n (t^{(n)} - x^{(n)}) z^{(n)} = \text{prediction error} \times \text{feature}$$

$$w \leftarrow w - \eta \frac{d}{dw} G(w) \quad \text{iteratively step down the objective (gradient points up hill)} \quad 39$$

Given the objective function, we should find the optimal ω that minimizes the objective function.

$$G(\omega) = - \sum_n [t^{(n)} \log (x(z^{(n)}; \omega)) + (1 - t^{(n)}) \log (1 - x(z^{(n)}; \omega))] \geq 0$$

First, we calculate the derivative of $\log(x(z^{(n)}; \omega))$, and $x(\cdot)$ is a sigmoid function, $x(z) = \frac{1}{1+e^{-z}}$, $z^{(n)} = \sum_i \omega_i z_i$,

$$\begin{aligned}
\frac{d}{d\omega_i} \log(\sigma(z^{(n)}; \omega)) &= \frac{d \log(x(z^{(n)}; \omega))}{dz^{(n)}} \frac{dz^{(n)}}{d\omega_i} \\
\frac{dz^{(n)}}{d\omega_i} &= z_i \\
\frac{d \log(x(z^{(n)}; \omega))}{dz^{(n)}} &= \frac{1}{x(z^{(n)}; \omega)} \frac{dx(z^{(n)}; \omega)}{dz^{(n)}} \\
&= \frac{1}{x(z^{(n)}; \omega)} x(z^{(n)}; \omega) (1 - x(z^{(n)}; \omega)) \\
&= 1 - x(z^{(n)}; \omega) \\
\frac{d}{d\omega_i} (1 - \log(\sigma(z^{(n)}; \omega))) &= \frac{d(1 - \log(x(z^{(n)}; \omega)))}{dz^{(n)}} \frac{dz^{(n)}}{d\omega_i} \\
\frac{d(1 - \log(x(z^{(n)}; \omega)))}{dz^{(n)}} &= -\frac{1}{1 - x(z^{(n)}; \omega)} \frac{dx(z^{(n)}; \omega)}{dz^{(n)}} \\
&= -\frac{1}{1 - x(z^{(n)}; \omega)} x(z^{(n)}; \omega) (1 - x(z^{(n)}; \omega)) \\
&= -x(z^{(n)}; \omega)
\end{aligned}$$

So, we can obtain the result of $\frac{d}{d\omega_i} G(\omega)$

$$\begin{aligned}
\frac{d}{d\omega_i} G(\omega) &= -\frac{d}{d\omega_i} \sum_n [t^{(n)} \log(x(z^{(n)}; \omega)) + (1 - t^{(n)}) \log(1 - x(z^{(n)}; \omega))] \\
&= -\sum_n [t^{(n)} (1 - x(z^{(n)}; \omega)) z_i^{(n)} - (1 - t^{(n)}) x(z^{(n)}; \omega) z_i^{(n)}] \\
&= -\sum_n [t^{(n)} - (x(z^{(n)}; \omega))] z_i^{(n)} = -\sum_n [t^{(n)} - x_i^{(n)}] z_i^{(n)}
\end{aligned}$$

Finally, we can get the result of $\frac{d}{d\omega} G(\omega)$

$$\frac{d}{d\omega} G(\omega) = -\sum_n [t^{(n)} - x^{(n)}] z^{(n)}$$