Please use this report template, and upload it in the PDF format. Reports in other forms/formats will result in ZERO point. Reports written in either Chinese or English is acceptable. The length of your report should NOT exceed 6 pages (**excluding bonus**).

Name:鍾勝隆 Dep.:電信碩一 Student ID:R06942052

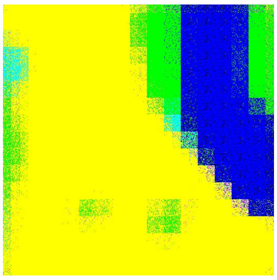
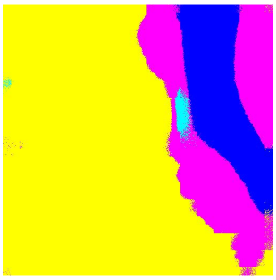
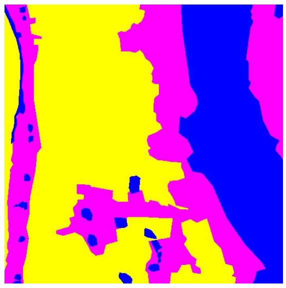
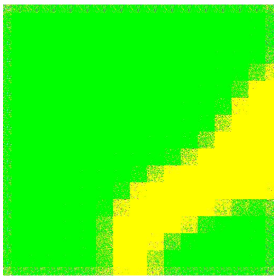
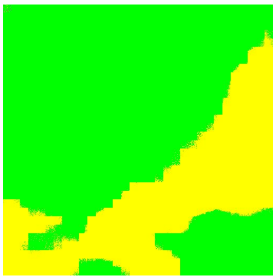
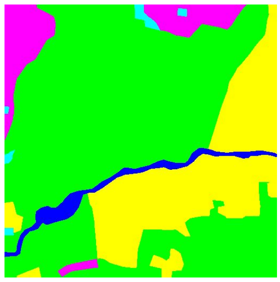
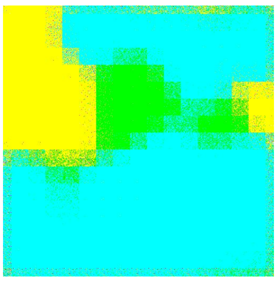
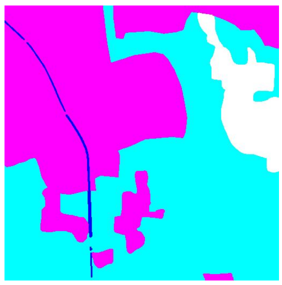
1. (5%) Print the network architecture of your VGG16-FCN32s model.

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
conv2d_1 (Conv2D)	(None, 16, 16, 4096)	18878464
dropout_1 (Dropout)	(None, 16, 16, 4096)	0

conv2d_2 (Conv2D)	(None, 16, 16, 4096)	16781312
dropout_2 (Dropout)	(None, 16, 16, 4096)	0
conv2d_3 (Conv2D)	(None, 16, 16, 7)	28679
FCN_convtrans1 (Conv2DTranspose)	(None, 512, 512, 7)	200711
activation_1 (Activation)	(None, 512, 512, 7)	0

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)

   The followings are my FCN_32s Model result of 1st, 15th, 30th epoch

	1st epoch	15th epoch	30th epoch	Ground Truth
0008_mask				
0097_mask				
0107_mask				

3. (15%) Implement an improved model which performs better than your baseline model. Print the network architecture of this model.

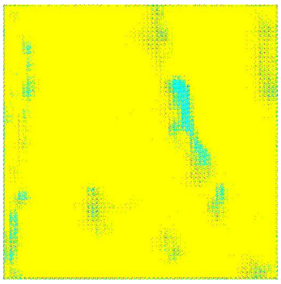
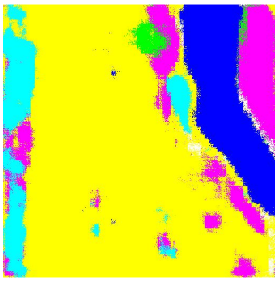
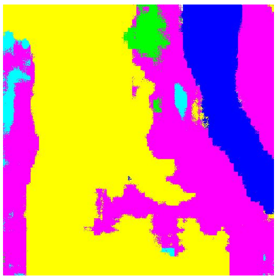
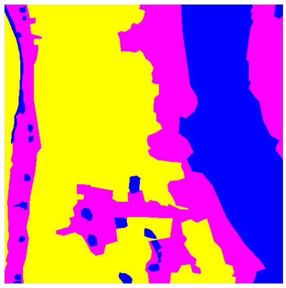
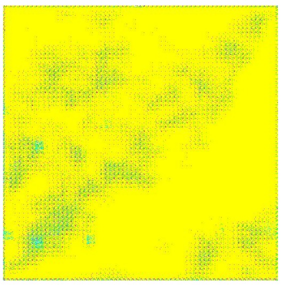
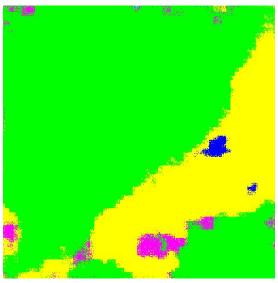
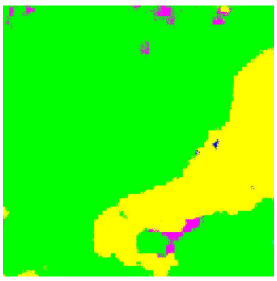
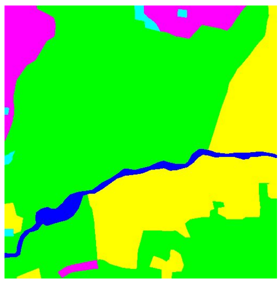
I implemented the **FCN-8s** as the improved model.

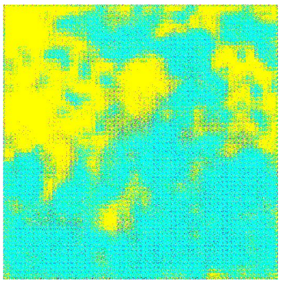
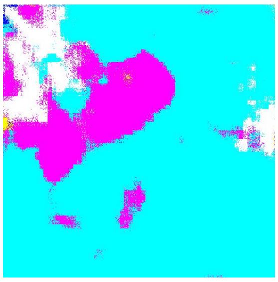
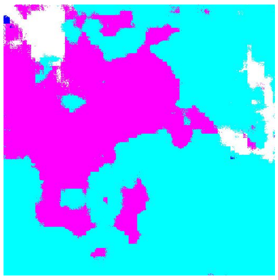
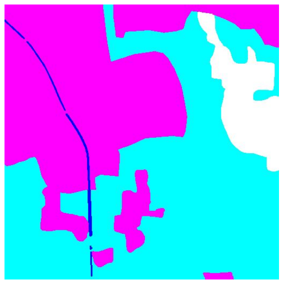
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 512, 512, 3)	0	
block1_conv1 (Conv2D)	(None, 512, 512, 64)	1792	input_1[0][0]
block1_conv2 (Conv2D)	(None, 512, 512, 64)	36928	block1_conv1[0][0]
block1_pool (MaxPooling2D)	(None, 256, 256, 64)	0	block1_conv2[0][0]
block2_conv1 (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]
conv2d_1 (Conv2D)	(None, 16, 16, 4096)	18878464	block5_pool[0][0]
dropout_1 (Dropout)	(None, 16, 16, 4096)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 16, 16, 4096)	16781312	dropout_1[0][0]
dropout_2 (Dropout)	(None, 16, 16, 4096)	0	conv2d_2[0][0]
conv2d_3 (Conv2D)	(None, 16, 16, 7)	28679	dropout_2[0][0]
conv2d_4 (Conv2D)	(None, 32, 32, 7)	3591	block4_pool[0][0]

FCN_Upsampling_Conv7 (Conv2DTranspose)	(None, 64, 64, 7)	3143	conv2d_3[0][0]
FCN_Upsampling_Pool4 (Conv2DTranspose)	(None, 64, 64, 7)	791	conv2d_4[0][0]
conv2d_5 (Conv2D)	(None, 64, 64, 7)	1799	block3_pool[0][0]
add_1 (Add)	(None, 64, 64, 7)	0	FCN_Upsampling_Conv7[0][0] FCN_Upsampling_Pool4[0][0] conv2d_5[0][0]
FCN_convtrans3 (Conv2DTranspose)	(None, 512, 512, 7)	12551	add_1[0][0]
activation_1 (Activation)	(None, 512, 512, 7)	0	FCN_convtrans3[0][0]

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.

   The followings are my FCN_8s Model result of 1st, 20th, 40th epoch

	1st epoch	20th epoch	40th epoch	Ground Truth
0008_mask				
0097_mask				

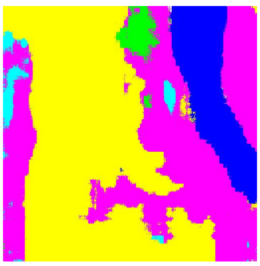
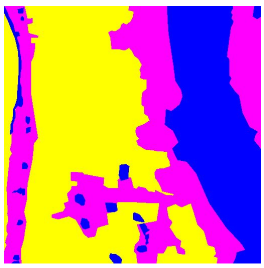
0107_mask				

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.

	VGG16-FCN32s Model	Improved Model
class #0	75.045%	**76.171**%
class #1	86.839%	**88.171**%
class #2	33.729%	**34.525**%
class #3	79.793%	**80.728**%
class #4	73.406%	**76.133**%
class #5	67.723%	**69.553**%
Mean_IOU	69.422%	**70.883**%

The improved model I used is the VGG16-FCN8s which will fuse the coarse to fine information together to train the FCN network. It makes the neural network find the detail because the maxpooling process will lose some details. Thus, adding the results of the different level maxpooling keeps the detail of the original satellite images.

We can find the evidences from the 0008_mask of both the models. FCN-8s can do the finer classification at edge of different class.

FCN-32s	FCN-8s	Ground Truth
		

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

objective function:

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

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

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}} G(\boldsymbol{w}) = \sum_n \frac{\mathrm{d}G(\boldsymbol{w})}{\mathrm{d}x^{(n)}} \frac{\mathrm{d}x^{(n)}}{\mathrm{d}\boldsymbol{w}} = -\sum_n (t^{(n)} - x^{(n)}) \boldsymbol{z}^{(n)}$$ = prediction error x feature

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

$x(z^{(n)}; w)$ is the sigmoid function. Derivative of the sigmoid function is $x(z^{(n)}; w)(1 - x(z^{(n)}; w))$

$$\frac{d}{dw} G(w) = \; -\Sigma_n \frac{dG(w)}{dx^{(n)}} \frac{dx^{(n)}}{dw}$$

$$= \; -\Sigma_n \frac{d(\, t^{(n)} ln\, x(z^{(n)}; w) + (1 - t^{(n)}) ln\, (1 - x(z^{(n)}; w))\,)}{dx^{(n)}} \frac{dx^{(n)}}{dw}$$

$$\frac{d\, ln\, x(z^{(n)}; w)}{d\, x^{(n)}} = \frac{1}{x(z^{(n)}; w)} \frac{d\, x(z^{(n)}; w)}{d\, x^{(n)}} = \frac{1}{x(z^{(n)}; w)} x(z^{(n)}; w)(1 - x(z^{(n)}; w)) = 1 - x(z^{(n)}; w)$$

$$\frac{d\, ln\, (1 - x(z^{(n)}; w))}{d\, x^{(n)}} = \frac{1}{1 - x(z^{(n)}; w)} \frac{d\, x(z^{(n)}; w)}{d\, x^{(n)}} = \; - \frac{1}{1 - x(z^{(n)}; w)} x(z^{(n)}; w)(1 - x(z^{(n)}; w)) = \; -x(z^{(n)}; w)$$

$$-\Sigma_n \frac{d(\, t^{(n)} ln\, x(z^{(n)}; w) + (1 - t^{(n)}) ln\, (1 - x(z^{(n)}; w))\,)}{dx^{(n)}} \frac{dx^{(n)}}{dw}$$

$$= \; -\Sigma_n\, t^{(n)}(1 - x(z^{(n)}; w)) + (\, 1 - t^{(n)})(-\, x(z^{(n)}; w)) \cdot z^{(n)}$$

$$= \; -\Sigma_n\, (t^{(n)} - x(z^{(n)}; w)) \cdot z^{(n)}$$

$$\frac{d}{dw} G(w) = \; -\Sigma_n\, (t^{(n)} - x(z^{(n)}; w)) \cdot z^{(n)} \quad \text{Proved.}$$

Reference:

1. FCN-32s sample code
   https://github.com/divamgupta/image-segmentation-keras

2. J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.