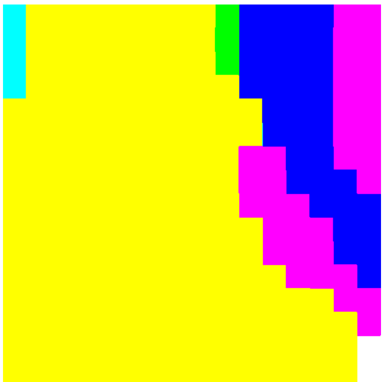
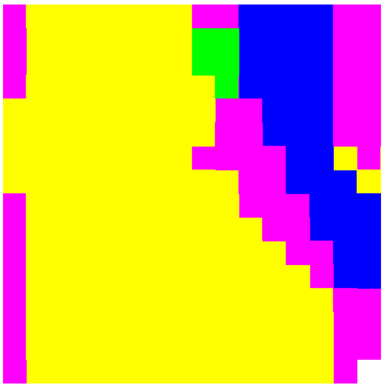
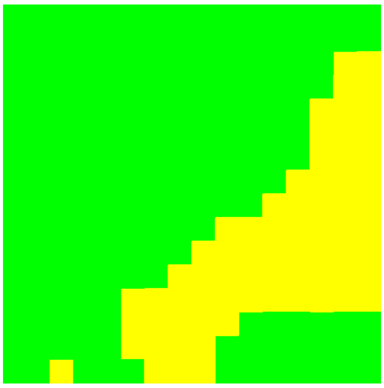
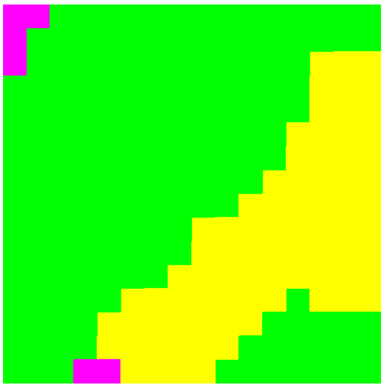
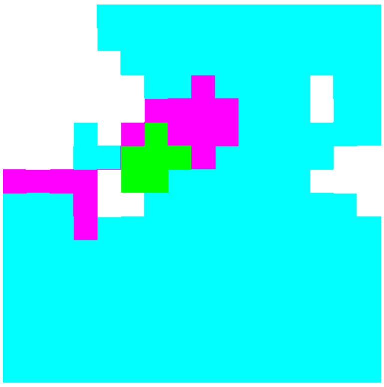
Please use this report template, and upload it in the PDF format. Reports in other forms/formats will result in ZERO point. Reports written in either Chinese or English is acceptable. The length of your report should NOT exceed 6 pages (**excluding bonus**).

Name: 何明倩 Dep.:電信碩一 Student ID:R06942039

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_up_sampling (UpSampli (None, 512, 512, 512)     0
_________________________________________________________________
block6_conv (Conv2D)         (None, 512, 512, 7)       14343
=================================================================
Total params: 14,729,031
Trainable params: 14,729,031
Non-trainable params: 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)

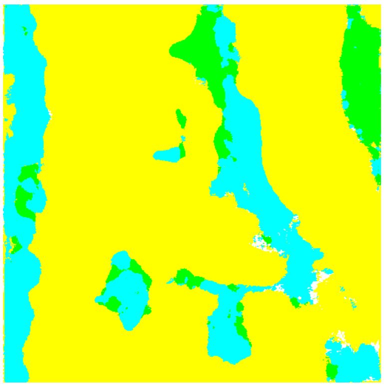
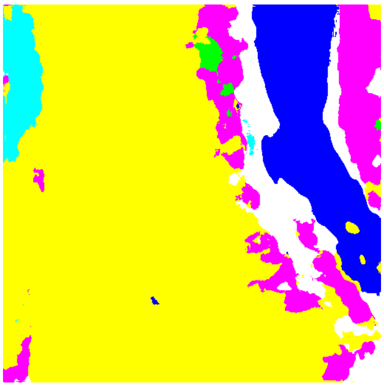
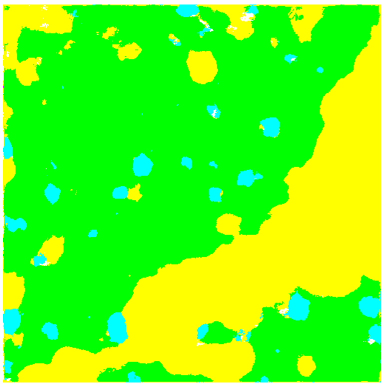
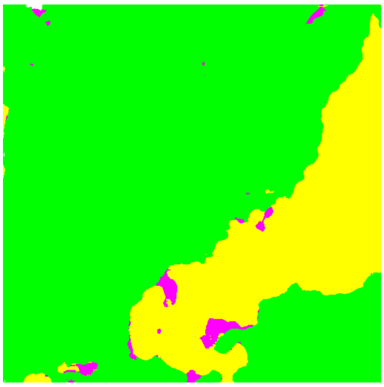
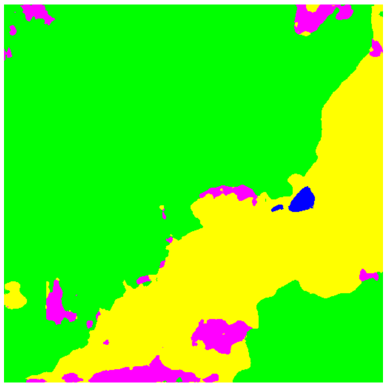
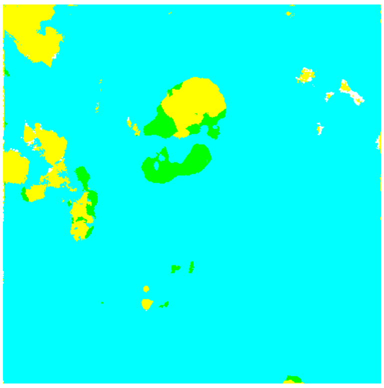
	Early_00	Middle_37	Final_64
0008_sat.jpg			
0097_sat.jpg			
0107_sat.jpg			

3. (15%) Implement an improved model which performs better than your baseline model. Print the network architecture of this 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]
block6_up (UpSampling2D)	(None, 64, 64, 512)	0	block5_conv3[0][0]
block6_conv1 (Conv2D)	(None, 64, 64, 512)	1049088	block6_up[0][0]
block6_concat (Concatenate)	(None, 64, 64, 1024)	0	block4_conv3[0][0] block6_conv1[0][0]
block6_conv2 (Conv2D)	(None, 64, 64, 512)	4719104	block6_concat[0][0]
block6_conv3 (Conv2D)	(None, 64, 64, 512)	2359808	block6_conv2[0][0]
block7_up (UpSampling2D)	(None, 128, 128, 512)	0	block6_conv3[0][0]
block7_conv1 (Conv2D)	(None, 128, 128, 256)	524544	block7_up[0][0]
block7_concat (Concatenate)	(None, 128, 128, 512)	0	block3_conv3[0][0] block7_conv1[0][0]
block7_conv2 (Conv2D)	(None, 128, 128, 256)	1179904	block7_concat[0][0]
block7_conv3 (Conv2D)	(None, 128, 128, 256)	590080	block7_conv2[0][0]
block8_up (UpSampling2D)	(None, 256, 256, 256)	0	block7_conv3[0][0]
block8_conv1 (Conv2D)	(None, 256, 256, 128)	131200	block8_up[0][0]
block8_concat (Concatenate)	(None, 256, 256, 256)	0	block2_conv2[0][0] block8_conv1[0][0]
block8_conv2 (Conv2D)	(None, 256, 256, 128)	295040	block8_concat[0][0]
block8_conv3 (Conv2D)	(None, 256, 256, 128)	147584	block8_conv2[0][0]
block9_up (UpSampling2D)	(None, 512, 512, 128)	0	block8_conv3[0][0]
block9_conv1 (Conv2D)	(None, 512, 512, 64)	32832	block9_up[0][0]
block9_concat (Concatenate)	(None, 512, 512, 128)	0	block1_conv2[0][0] block9_conv1[0][0]
block9_conv2 (Conv2D)	(None, 512, 512, 64)	73792	block9_concat[0][0]
block9_conv3 (Conv2D)	(None, 512, 512, 64)	36928	block9_conv2[0][0]
block10_conv1 (Conv2D)	(None, 512, 512, 16)	9232	block9_conv3[0][0]
predictions (Conv2D)	(None, 512, 512, 7)	119	block10_conv1[0][0]

Total params: 25,863,943
Trainable params: 25,863,943
Non-trainable params: 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.

	Early_00	Middle_11	Final_24
0008_sat.jpg			
0097_sat.jpg			
0107_sat.jpg			

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.

	FCNN32	U_net [1]	U_net_VGG16_based
mIoU	0.6711	0.6386	0.6875

Table 1 各個 model 的 mIOU

我嘗試使用 FCNN 和 U_net。

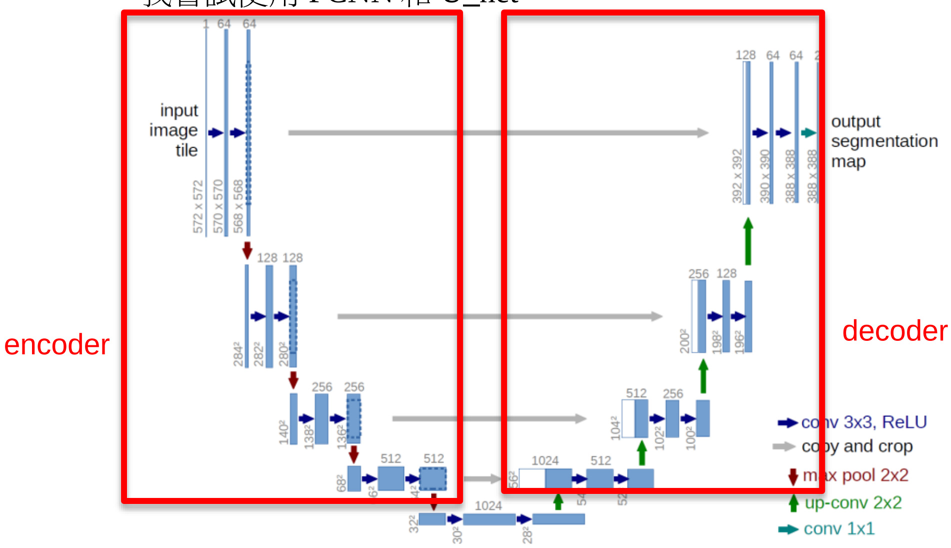


figure 1 U_net architecture

根據圖一的 U_net 架構，在 model 架構中會將 encode 的資訊 concatenate 到 decoder 端來增加 spatial information，感覺會比 FCNN32 加入更多 spatial information，會 predict 越精確。只是根據 table 1 可以發現如果直接用原本的 U_net model[1]的結果並沒有比 FCNN32 performance 好，我猜測是不是因為 FCNN32 還有用到 pre-train weight 會讓他找出更好的 features 來決定每個 pixel 是屬於哪個 class。因此我嘗試將 VGG16 最後一層 maxpooling 之前的 layers 拿來當 U_net 的 encoder 部份，希望可以藉由 pre-train 的 weight 協助 model 取到更好的 features。而根據 table 1 的結果可以明顯看出 U_net_VGG16_based 的 model 就比 FCNN32 高出了 0.016%，由此可以推斷出 U_net 將 encoder 的資訊加入 decoder 中是可以提升他預測的 precision，而 pre-train 的 weight 可以協助找出更好的 feature 進行預測，這也是為什麼我用 VGG16 based 的 U_net performance 會有所提升的原因。

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

objective function:

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

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w})$$ 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