Name: 藍聖皓 Dep.:電信碩一 Student ID:R06942143

討論：林恩妤、簡上淵、洪健豪

參考資源：https://github.com/divamgupta/image-segmentation-keras/tree/master/Models

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

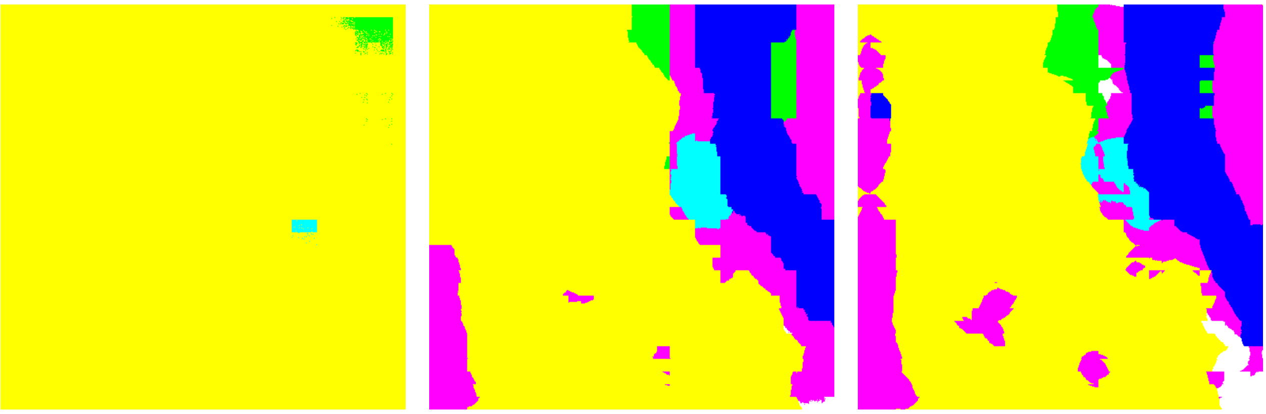
```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 512, 512, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 512, 512, 64)      1792
_________________________________________________________________
activation_1 (Activation)    (None, 512, 512, 64)      0
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 512, 512, 64)      36928
_________________________________________________________________
activation_2 (Activation)    (None, 512, 512, 64)      0
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 256, 256, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 256, 256, 128)     73856
_________________________________________________________________
activation_3 (Activation)    (None, 256, 256, 128)     0
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 256, 256, 128)     147584
_________________________________________________________________
activation_4 (Activation)    (None, 256, 256, 128)     0
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 128, 128, 128)     0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 128, 128, 256)     295168
_________________________________________________________________
activation_5 (Activation)    (None, 128, 128, 256)     0
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 128, 128, 256)     590080
_________________________________________________________________
activation_6 (Activation)    (None, 128, 128, 256)     0
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 128, 128, 256)     590080
_________________________________________________________________
activation_7 (Activation)    (None, 128, 128, 256)     0
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 64, 64, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 64, 64, 512)       1180160
_________________________________________________________________
activation_8 (Activation)    (None, 64, 64, 512)       0
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 64, 64, 512)       2359808
_________________________________________________________________
activation_9 (Activation)    (None, 64, 64, 512)       0
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 64, 64, 512)       2359808
_________________________________________________________________
   activation_10 (Activation)   (None, 64, 64, 512)       0
_________________________________________________________________
block4_pool   (MaxPooling2D)       (None,   32,   32,   512)
0_________________________
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
activation_11 (Activation)   (None, 32, 32, 512)       0
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
activation_12 (Activation)   (None, 32, 32, 512)       0
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
activation_13 (Activation)   (None, 32, 32, 512)       0
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 16, 16, 512)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 2048)      51382272
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 2048)      8192
_________________________________________________________________
activation_14 (Activation)   (None, 16, 16, 2048)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 2048)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 2048)      4196352
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 2048)      8192
_________________________________________________________________
activation_15 (Activation)   (None, 16, 16, 2048)      0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 2048)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 7)         14343
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 7)         28
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 512, 512, 7)       200711
=================================================================
Total params: 70,524,778
Trainable params: 58,161,692
Non-trainable params: 12,363,086
_________________________________________________________________
```

前面也使用了 VGG16 的架構，這邊除了 VGG16 的最後一層開成 trainable 之外，其餘的都是不能更動的參數(因為原本實驗的時候，把全部都開成 trainable，因此最後 overfitting 在 training set 上面(loss 0.22，acc = 0.95)，而 validation 的分數就很差(loss = 1.05，acc = 0.76)，於是做了這樣的調整，接下來接了三層的 fully connected cnn 之後做 transpose，這邊都與論文上的架構一樣。不過後來有再調整在每個 fully connected 之間加入了 batch normalize，為了使其標準化，提高學習速率。

Type(# of epoch)	All trainable(100)	VGG last trainable(40)	Batch normalize(40)
Train loss	0.124	0.224	0.132
Train acc	0.95	0.923	0.945
Validation loss	1.05	0.543	0.49
Validation acc	0.76	0.843	0.86

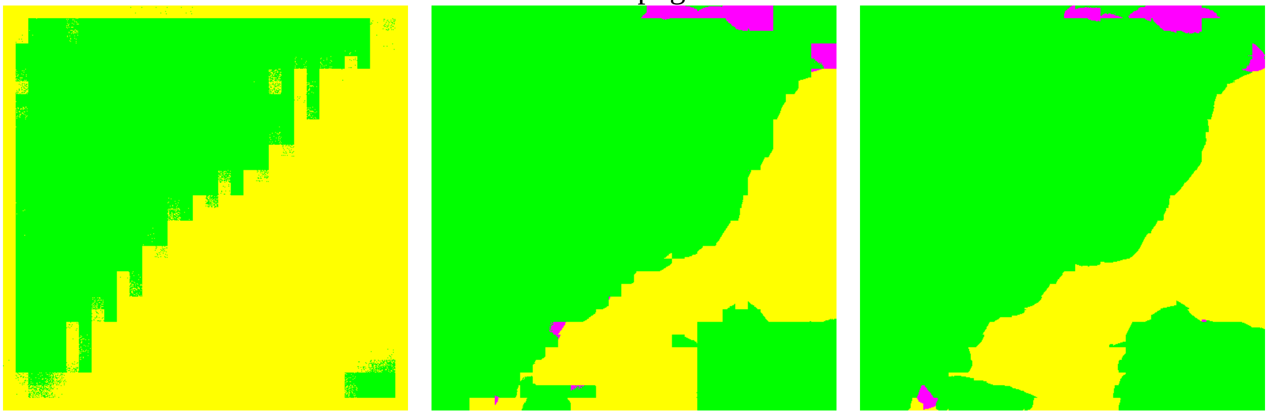
2. (10%) Show the predicted segmentation mask of validation/0008_sat.jpg, validation/0097_sat.jpg, validation/0107_sat.jpg during the early, middle, and the final stage during the training stage. (For example, results of 1st, 10th, 20th epoch)

0008.png



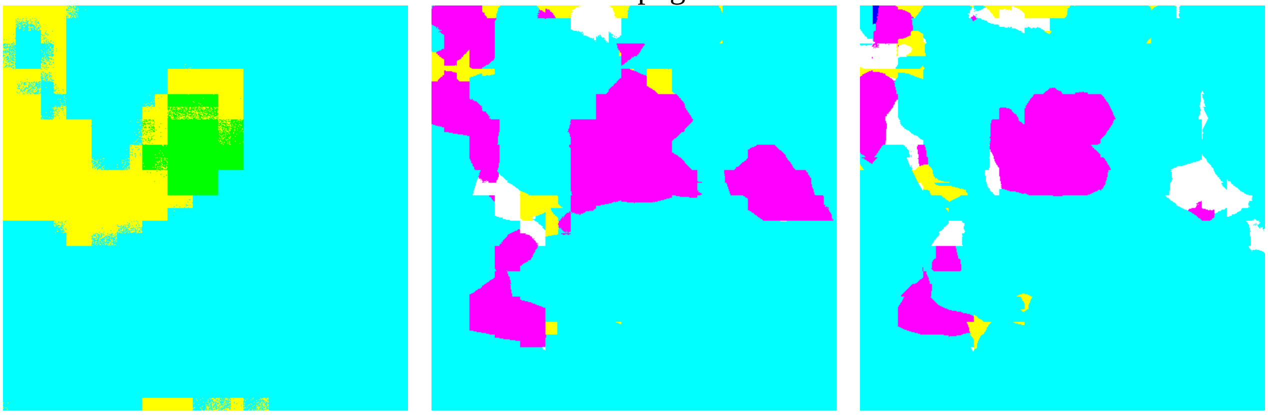
early middle final one

0097.png



early middle final one

0107.png



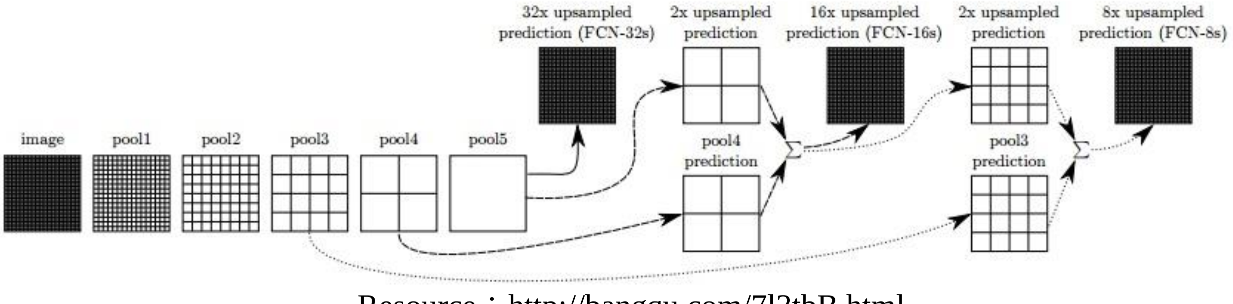
early middle final one

可以從這些圖中看出，在 training 的過程中，可以慢慢的區隔出更多的區塊，不過 FCN32 所分出來的區塊，好似都比較方方正正，對於比較細部的資訊就沒有那麼能分辨，這也可以從 IOU 中看出來，對於 case#2(藍色、河流)的分辨效果就很低，因為河流的資訊大部分都是很細的一長條，可能這條資訊就會再多次的 conv 被忽略掉。

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]
____________________________________________________________
activation_1 (Activation)      (None, 512, 512, 64) 0          block1_conv1[0][0]
____________________________________________________________
block1_conv2 (Conv2D)          (None, 512, 512, 64) 36928      activation_1[0][0]
____________________________________________________________
activation_2 (Activation)      (None, 512, 512, 64) 0          block1_conv2[0][0]
____________________________________________________________
block1_pool (MaxPooling2D)     (None, 256, 256, 64) 0          activation_2[0][0]
____________________________________________________________
block2_conv1 (Conv2D)          (None, 256, 256, 128 73856      block1_pool[0][0]
____________________________________________________________
activation_3 (Activation)      (None, 256, 256, 128 0          block2_conv1[0][0]
____________________________________________________________
block2_conv2 (Conv2D)          (None, 256, 256, 128 147584     activation_3[0][0]
____________________________________________________________
activation_4 (Activation)      (None, 256, 256, 128 0          block2_conv2[0][0]
____________________________________________________________
block2_pool (MaxPooling2D)     (None, 128, 128, 128 0          activation_4[0][0]
____________________________________________________________
block3_conv1 (Conv2D)          (None, 128, 128, 256 295168     block2_pool[0][0]
____________________________________________________________
activation_5 (Activation)      (None, 128, 128, 256 0          block3_conv1[0][0]
____________________________________________________________
block3_conv2 (Conv2D)          (None, 128, 128, 256 590080     activation_5[0][0]
____________________________________________________________
activation_6 (Activation)      (None, 128, 128, 256 0          block3_conv2[0][0]
____________________________________________________________
block3_conv3 (Conv2D)          (None, 128, 128, 256 590080     activation_6[0][0]
____________________________________________________________
activation_7 (Activation)      (None, 128, 128, 256 0          block3_conv3[0][0]
____________________________________________________________
block3_pool (MaxPooling2D)     (None, 64, 64, 256)  0          activation_7[0][0]
block4_conv1 (Conv2D)          (None, 64, 64, 512)  1180160    block3_pool[0][0]
____________________________________________________________
activation_8 (Activation)      (None, 64, 64, 512)  0          block4_conv1[0][0]
____________________________________________________________
block4_conv2 (Conv2D)          (None, 64, 64, 512)  2359808    activation_8[0][0]
____________________________________________________________
activation_9 (Activation)      (None, 64, 64, 512)  0          block4_conv2[0][0]
block4_conv3 (Conv2D)          (None, 64, 64, 512)  2359808    activation_9[0][0]
____________________________________________________________
activation_10 (Activation)     (None, 64, 64, 512)  0          block4_conv3[0][0]
____________________________________________________________
block4_pool (MaxPooling2D)     (None, 32, 32, 512)  0          activation_10[0][0]
____________________________________________________________
block5_conv1 (Conv2D)          (None, 32, 32, 512)  2359808    block4_pool[0][0]
____________________________________________________________
activation_11 (Activation)     (None, 32, 32, 512)  0          block5_conv1[0][0]
____________________________________________________________
block5_conv2 (Conv2D)          (None, 32, 32, 512)  2359808    activation_11[0][0]
____________________________________________________________
activation_12 (Activation)     (None, 32, 32, 512)  0          block5_conv2[0][0]
____________________________________________________________
block5_conv3 (Conv2D)          (None, 32, 32, 512)  2359808    activation_12[0][0]
____________________________________________________________
activation_13 (Activation)     (None, 32, 32, 512)  0          block5_conv3[0][0]
____________________________________________________________
block5_pool (MaxPooling2D)     (None, 16, 16, 512)  0          activation_13[0][0]
____________________________________________________________
conv2d_1 (Conv2D)              (None, 16, 16, 1024) 25691136   block5_pool[0][0]
____________________________________________________________
dropout_1 (Dropout)            (None, 16, 16, 1024) 0          conv2d_1[0][0]
____________________________________________________________
conv2d_2 (Conv2D)              (None, 16, 16, 1024) 1049600    dropout_1[0][0]
____________________________________________________________
dropout_2 (Dropout)            (None, 16, 16, 1024) 0          conv2d_2[0][0]
____________________________________________________________
conv2d_3 (Conv2D)              (None, 16, 16, 7)    7175       dropout_2[0][0]
____________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 32, 32, 7)   784        conv2d_3[0][0]
____________________________________________________________
conv2d_4 (Conv2D)              (None, 32, 32, 7)    3591       block4_pool[0][0]
____________________________________________________________
add_1 (Add)                    (None, 32, 32, 7)    0          conv2d_transpose_1[0][0]
                                                                conv2d_4[0][0]
____________________________________________________________
conv2d_5 (Conv2D)              (None, 64, 64, 7)    1799       block3_pool[0][0]
____________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 64, 64, 7)   784        add_1[0][0]
____________________________________________________________
add_2 (Add)                    (None, 64, 64, 7)    0          conv2d_5[0][0]
                                                                conv2d_transpose_2[0][0]
____________________________________________________________
conv2d_transpose_3 (Conv2DTrans (None, 512, 512, 7)  12544     add_2[0][0]
____________________________________________________________
activation_14 (Activation)     (None, 512, 512, 7)  0          conv2d_transpose_3[0][0]
=================================================================
==============================
Total params: 41,482,101
Trainable params: 29,127,221
Non-trainable params: 12,354,880
_________________________________________________________________
____________________
Train on 2313 samples, validate on 257 samples
```

這邊的模型架構是使用 FCN8，把 f3 f4 f5pooling 之後的結果再跟 fully connected 做結合，如下圖：

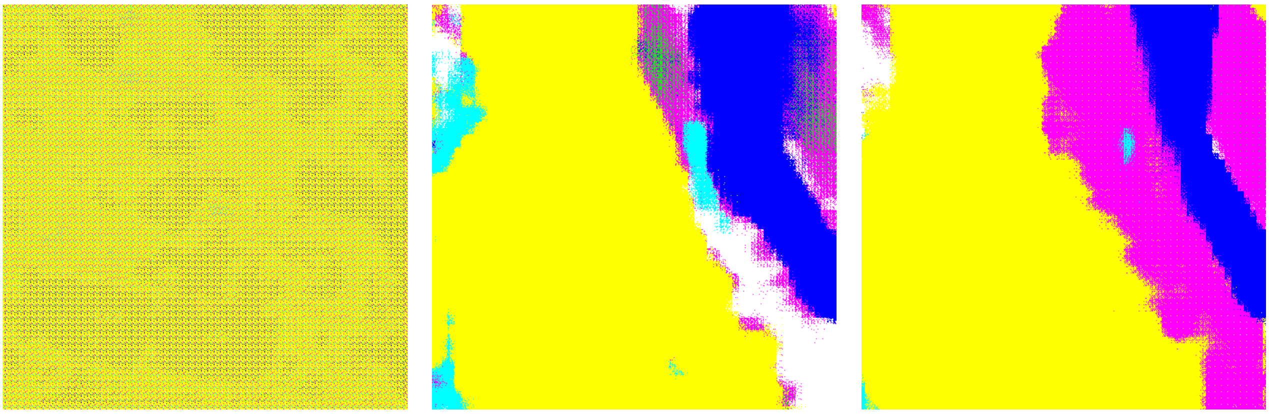


Resource：http://bangqu.com/7l3tbB.html

而這樣的架構有點類似 resnet，是將原本的 training data convolution 之後資訊，再加入到後面的 fully connected layer 中，這樣出來的結果會比較多保有原圖的資訊，而且對於細微的資訊也能抓到比較多，在下面第四題的結果可以看出來。

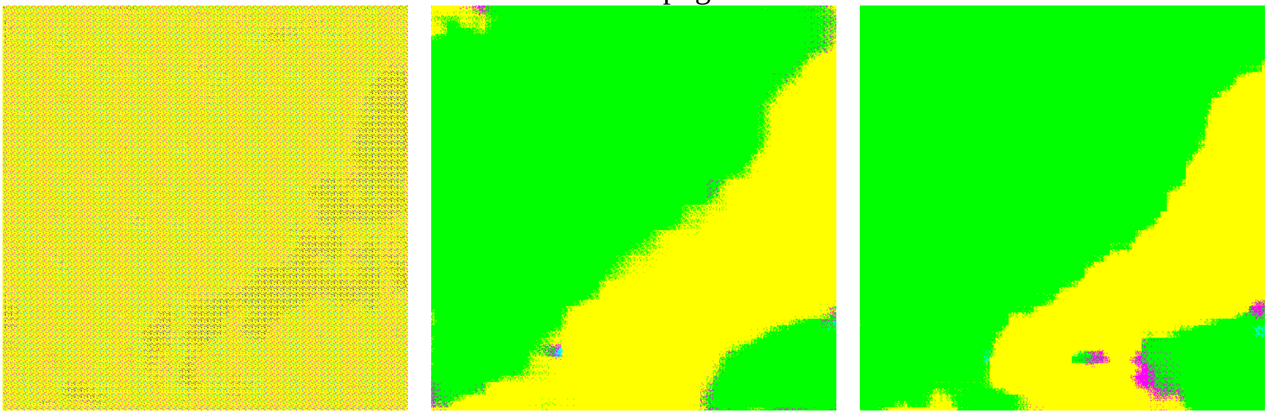
4. (10%) Show the predicted segmentation mask of validation/0008_sat.jpg, validation/0097_sat.jpg, validation/0107_sat.jpg during the early, middle, and the final stage during the training process of this improved model.

0008.png



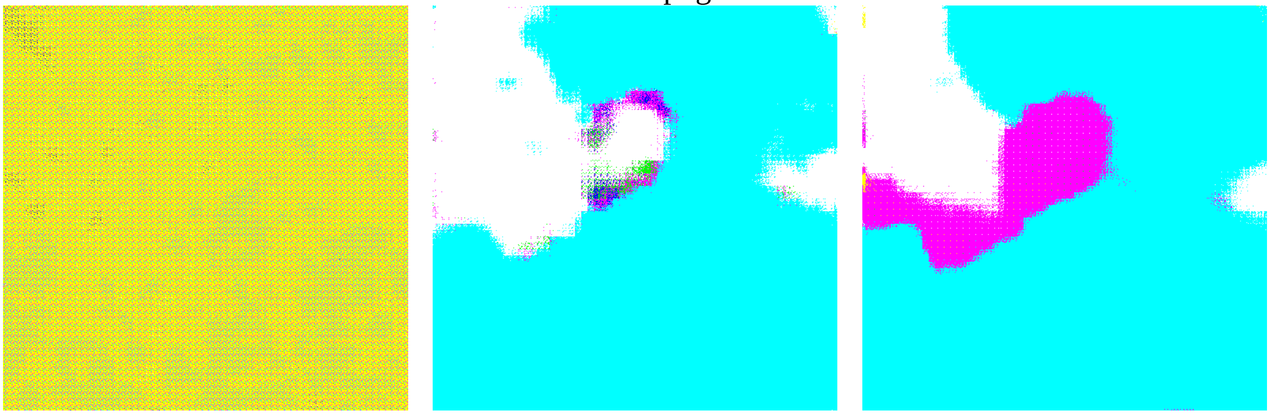
early middle final one

0097.png



early middle final one

0107.png

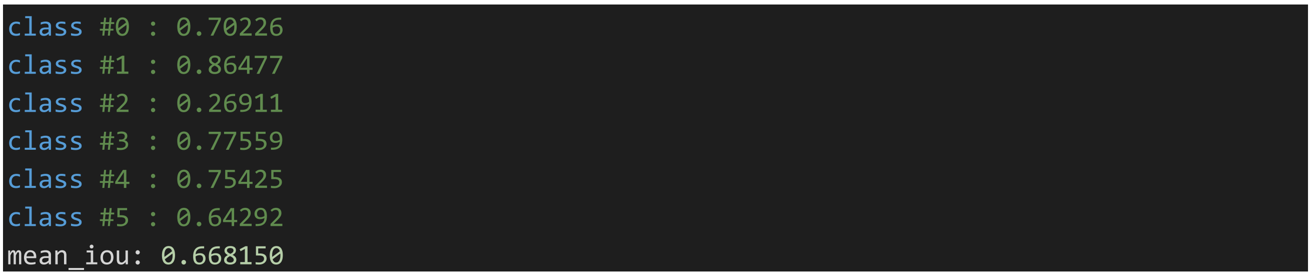


early middle final one

可以在 early 之中看到與 FCN32 有很大的不同，不過可以依稀看出來，這張圖中飽含了比較多的資訊(可能是原圖的細部資訊)，而且比起來更加的圓滑了些(不像 FCN32 這樣很方正)。

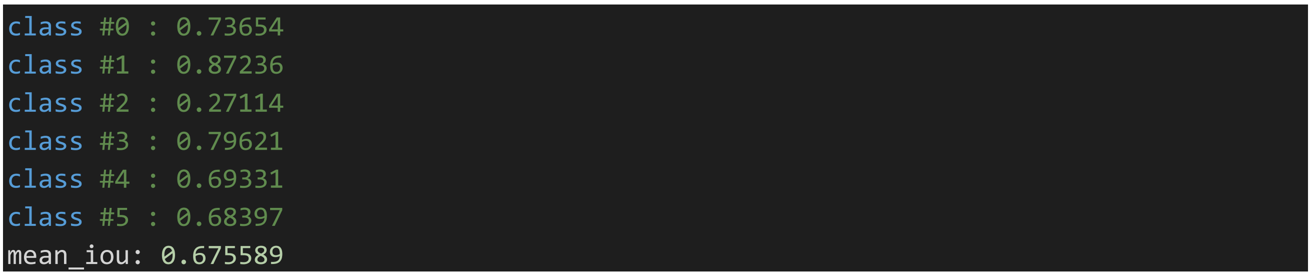
5. (15%) Report mIoU score of both models on the validation set. Discuss the reason why the improved model performs better than the baseline one. You may conduct some experiments and show some evidences to support your discussion.

FCN32



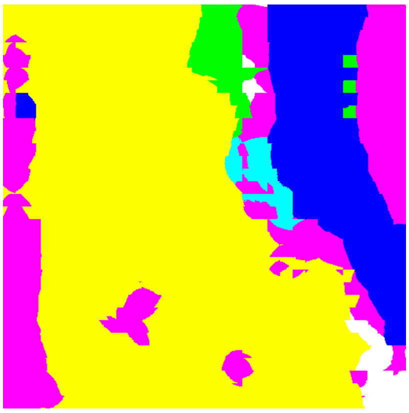
```
class #0 : 0.70226
class #1 : 0.86477
class #2 : 0.26911
class #3 : 0.77559
class #4 : 0.75425
class #5 : 0.64292
mean_iou: 0.668150
```

FCN8

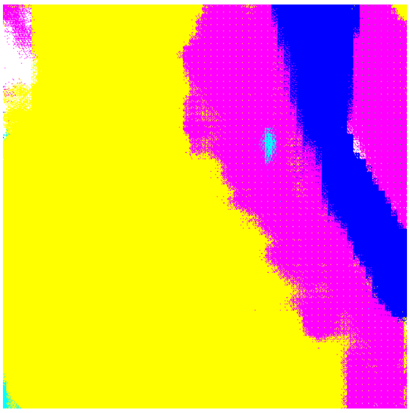


```
class #0 : 0.73654
class #1 : 0.87236
class #2 : 0.27114
class #3 : 0.79621
class #4 : 0.69331
class #5 : 0.68397
mean_iou: 0.675589
```

如同前面所講述的架構以及討論，FCN8 多加上了，f3、f4、f5 這幾層 pooling 出來的結果，和 transpose conv 相加，在丟到下一層，這樣加在一起之後，可以更多保有一些細部的資訊。可以從下面的圖中推得這件事情，FCN32 predict 出來的結果都比較像是一個 block 的感覺，而 FCN8 的結果就會比較細緻，可以看出預測結果的地方，FCN8 在邊界的地方處理較為模糊與圓潤，：



FCN32



FCN8

推測 FCN8 比 FCN32 在 mean_iou 上表現較好的原因：

1. 有更多資訊：

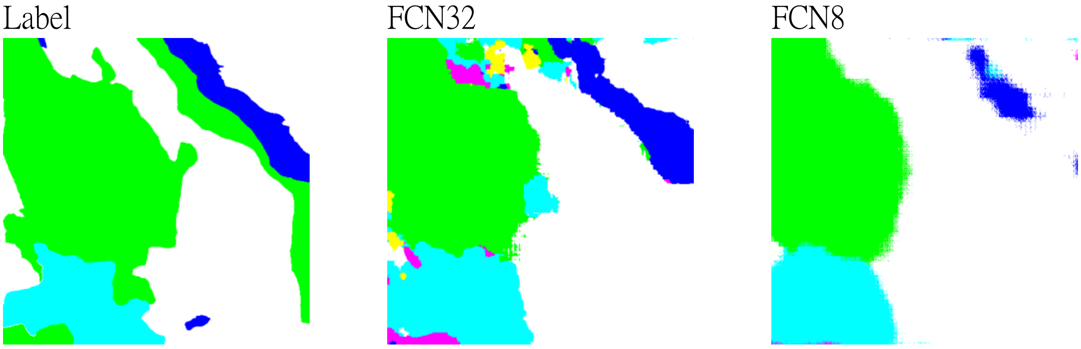
   FCN32 透過 fully connected conv，在每一層留下的資訊都慢慢的變少，transpose 回去之後，這失去的資訊都不見了，於是 transpose 的時候可能整塊都會被判別成黃色或是粉紅色，這樣子較細部的資料其實都是沒有被留存的，因此準確度較低。

   FCN8，透過很多層的 transpose conv，讓每層 transpose 回去之後，再把原本對應該層的 conv 資訊加回來，讓 transpose 回去時候，補足這些失去的資訊，重建的時候會更能貼切保有細部的資訊。

2. 結果計算方式：

   這邊也覺得和這樣的計算方式有關係，因為有去觀察實際上的比較結果，FCN32 在分

類結果的確沒有 FCN8 來的好，不過形狀倒是分類的不差(不過顏色分錯了，但是形狀整體看起來的效果較好)，也很符合剛剛前面說的推得的，而 mean_iou 結果較好的原因，是因為比較是比較該區塊的分類結果，FCN32 雖然形狀分類的比較好，但是卻因為形狀對了，裡面的顏色錯了，所以整體分數比較低，這邊如果量測結果的指標是有關注形狀的話，可能 FCN32 結果會好一些。


Label
FCN32
FCN8

這張圖是 242_mask.png，可以大略觀察出 FCN32 這張圖的形狀應該是比較好的，只是 segmentation 效果差了滿多的，因此最後 mean_iou 結果比較好。

結論：

加入了前面幾層的細部資訊，可以提高 segmentation 的結果(觀察起來被判別錯的區塊變少了 False negative 變少了)，不過 true positive 的數量也減少了一些，但是整體結果看起來比較好。

IOU = True Positive / (True Positive + False Positive + False Negative)

而會比較好的原因如上面兩點所述，剛好 FN 減少的量大於了 TP 減少的數量，因此 mean IOU 整體看下來效果是好的。

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

$$G(w) = -\sum_n \left[\hat{y}^{(n)} \log X(z^{(n)}; w) + (1-\hat{y}^{(n)}) \log(1 - X(z^{(n)}; w))\right]$$

$$\frac{dG(w)}{dX^{(n)}} \cdot \frac{dX^{(n)}}{dw} \qquad [\log X(z^{(n)})]' = [1 - X(z^{(n)})]\, z^{(n)}$$

$$= -\sum_n \left[\hat{y}^{(n)} [(1 - X(z^{(n)})] z^{(n)} + (1-\hat{y}^{(n)}) (-X(z^{(n)}) z^{(n)}\right]$$

$$= -\sum_n \left[\hat{y}^{(n)} (1 - X(z^{(n)}) + (1-\hat{y}^{(n)}) (-X(z^{(n)})\right] z^{(n)}$$

$$= -\sum_n \left[\hat{y}^{(n)} - \cancel{\hat{y}^{(n)} X(z^{(n)})} - X(z^{(n)}) + \cancel{\hat{y}^{(n)} X(z^{(n)})}\right] z^{(n)}$$

$$= -\sum_n \left[\hat{y}^{(n)} - X(z^{(n)})\right] z^{(n)}$$