HW2 Report

學號: r08942087 姓名: 吳彬睿

Problem 1: Image classification (10%)

1. (2%) Print the network of your model.

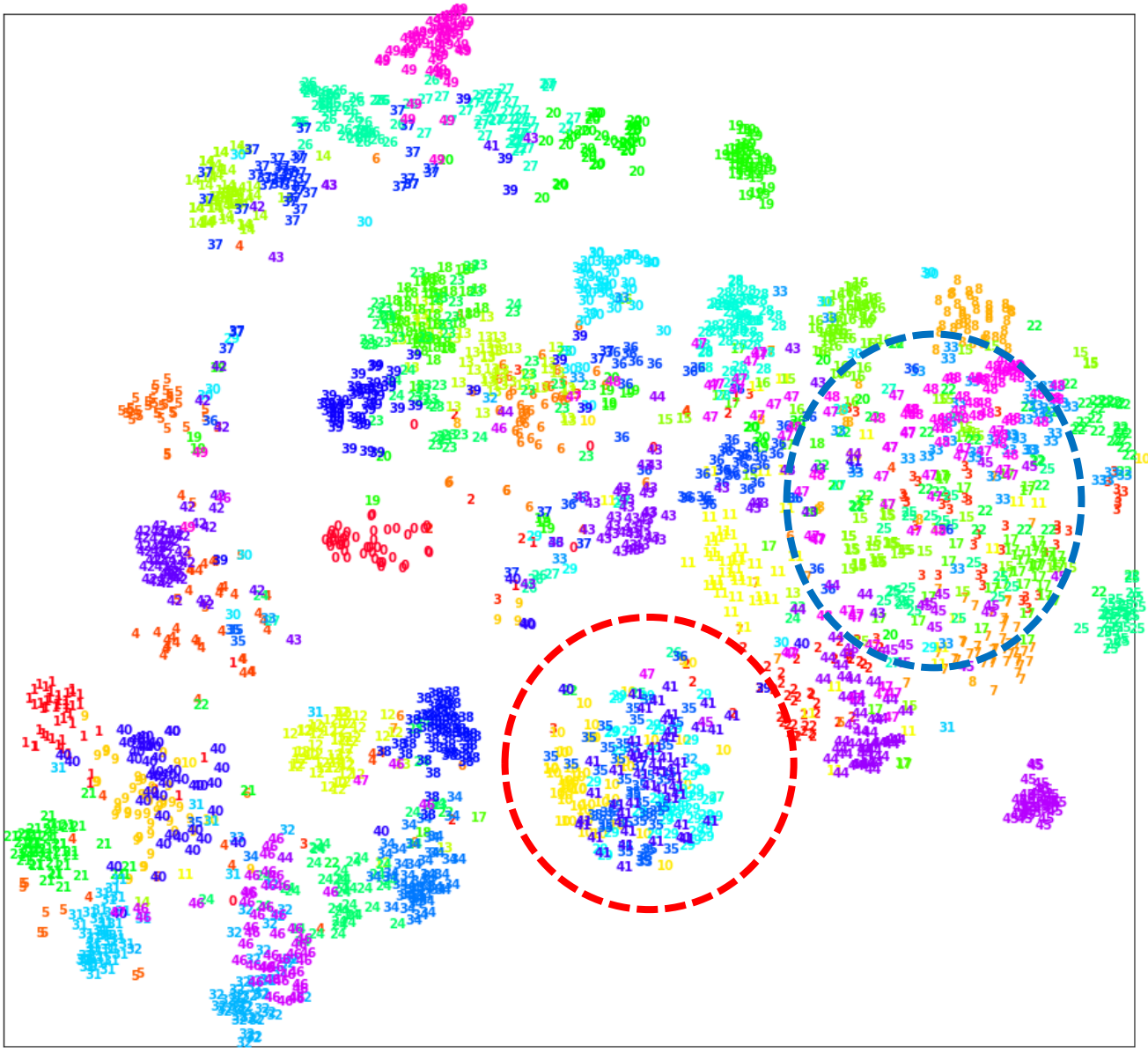
```
(features): Sequential(
    (0)   Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1)   ReLU(inplace=True)
    (2)   Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3)   ReLU(inplace=True)
    (4)   MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5)   Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6)   ReLU(inplace=True)
    (7)   Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8)   ReLU(inplace=True)
    (9)   MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10) Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11) ReLU(inplace=True)
    (12) Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13) ReLU(inplace=True)
    (14) Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15) ReLU(inplace=True)
    (16) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17) Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18) ReLU(inplace=True)
    (19) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20) ReLU(inplace=True)
    (21) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22) ReLU(inplace=True)
    (23) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25) ReLU(inplace=True)
    (26) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27) ReLU(inplace=True)
    (28) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29) ReLU(inplace=True)
    (30) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(fc): Seqential(
    (1)  Linear(8192, 50, bias=True)
)
```

2. (2%) Report accuracy of model on the validation set.

Valid accuracy: 0.776

3. (6%) Visualize the classification result on validation set by implementing t-SNE on output features of the second last layer. Briefly explain your result of tSNE visualization.

我們可以發現，不同類別的資料在倒數第二層有被切得比較開一點了，切得越開越有利於我們最後的分類。



切不太開的 classes (10:baby, 18:tree, 29:woman, 35:man, 41:girls)

這邊這一坨全部都是人的類別。

切不太開的 classes (3:rabbit, 15: raccoon, 22: skunk)

這邊這一坨全部都是小型動物的。

上面這兩群都是原本就長很像的東西，因此再降維度後他們本來就比較難切開。

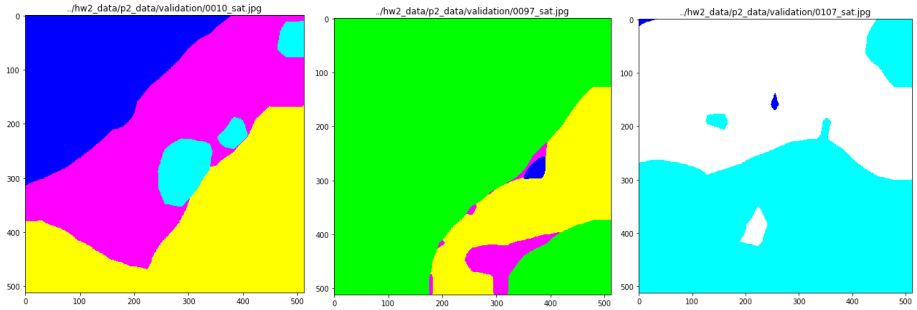
Problem 2: Semantic segmentation (30%)

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

```
(features): Sequential(
    (31)    Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (32)    ReLU(inplace=True)
    (33)    Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (34)    ReLU(inplace=True)
    (35)    MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (36)    Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (37)    ReLU(inplace=True)
    (38)    Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (39)    ReLU(inplace=True)
    (40)    MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (41) Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (42) ReLU(inplace=True)
    (43) Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (44) ReLU(inplace=True)
    (45) Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (46) ReLU(inplace=True)
    (47) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (48) Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (49) ReLU(inplace=True)
    (50) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (51) ReLU(inplace=True)
    (52) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (53) ReLU(inplace=True)
    (54) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (55) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (56) ReLU(inplace=True)
    (57) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (58) ReLU(inplace=True)
    (59) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (60) ReLU(inplace=True)
    (61) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(fc): fcn32(
    (1)  Conv2d(512, 4096, kernel_size=(7, 7), stride=(1, 1))
    (2)  ReLU(inplace=True)
    (3)  Dropout2d(p=0.5, inplace=False)
    (4)  Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
    (5)  ReLU(inplace=True)
    (6)  Dropout2d(p=0.5, inplace=False)
    (7)  Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
)
 (upscore): ConvTranspose2d(7, 7, kernel_size=(64, 64), stride=(32, 32), bias=False)
```

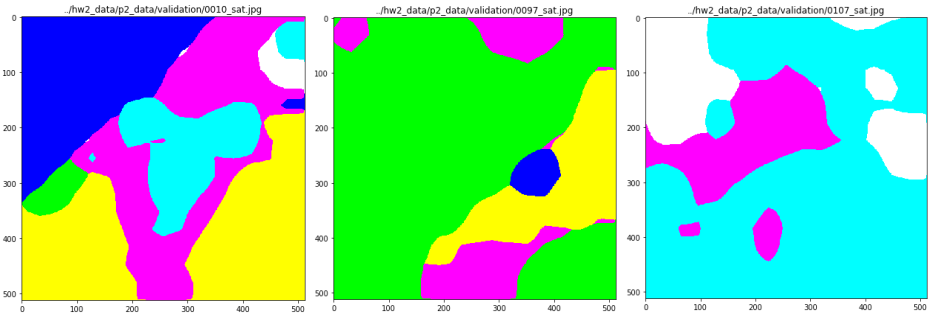
2. (5%) Show the predicted segmentation mask of “validation/0010_sat.jpg”, “validation/0097_sat”,” validation/0107_sat.jpg” during the early, middle, and the final stage during the training stage. (For example, results of 1st, 10st ,20st ,epoch)

- 1 epoch



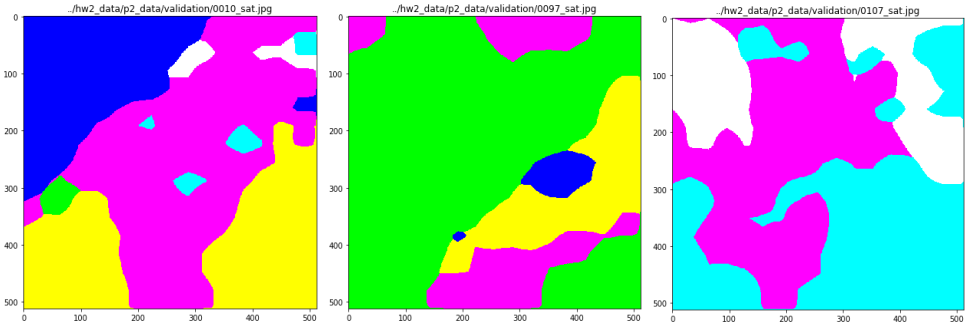
(0010_sat) (0097_sat) (0107_sat)

- 10 epoch



(0010_sat) (0097_sat) (0107_sat)

- 20 epoch



(0010_sat) (0097_sat) (0107_sat)

- Result

(0010_sat) (0097_sat) (0107_sat)

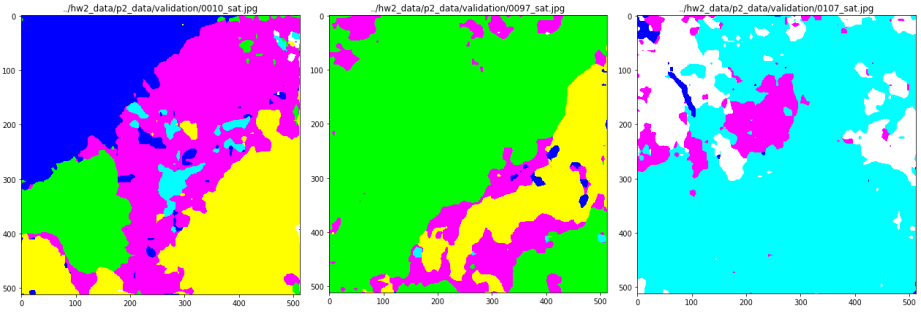
3. (5%) Implement an improved model which performs better than your baseline model. Print the network architecture of this model.

```
(features): Sequential(
  (0)   Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1)   ReLU(inplace=True)
    (2)   Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3)   ReLU(inplace=True)
    (4)   MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5)   Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6)   ReLU(inplace=True)
    (7)   Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8)   ReLU(inplace=True)
    (9)   MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10) Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11) ReLU(inplace=True)
    (12) Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13) ReLU(inplace=True)
    (14) Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15) ReLU(inplace=True)
    (16) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17) Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18) ReLU(inplace=True)
    (19) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20) ReLU(inplace=True)
    (21) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22) ReLU(inplace=True)
    (23) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25) ReLU(inplace=True)
    (26) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27) ReLU(inplace=True)
    (28) Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29) ReLU(inplace=True)
    (30) MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(fc): fcn32(
    (1)  Conv2d(512, 4096, kernel_size=(7, 7), stride=(1, 1))
    (2)  ReLU(inplace=True)
    (3)  Dropout2d(p=0.5, inplace=False)
    (4)  Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
    (5)  ReLU(inplace=True)
    (6)  Dropout2d(p=0.5, inplace=False)
)
(score_pool3): Conv2d(256, 7, kernel_size=(1, 1), stride=(1, 1), bias=False)
(score_pool4): Conv2d(512, 7, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

(score_fr): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1), bias=False)
(upscore2): ConvTranspose2d(7, 7, kernel_size=(4, 4), stride=(2, 2), bias=False)
(upscore8): ConvTranspose2d(7, 7, kernel_size=(16, 16), stride=(8, 8), bias=False)
(upscore_pool4): ConvTranspose2d(7, 7, kernel_size=(4, 4), stride=(2, 2), bias=False)

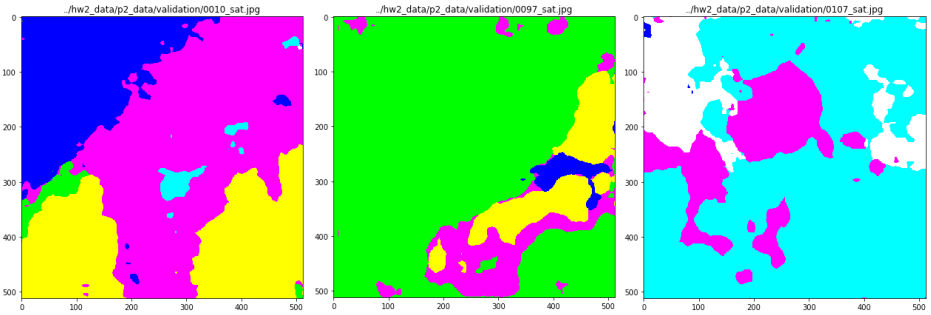
4. (5%) Show the predicted segmentation mask of of validation/0010_sat.jpg”, “validation/0097_sat.jpg”, “validation/0107_sat.jpg” during the early, middle, and the final stage during the training process of this improved model.

- 1 epoch



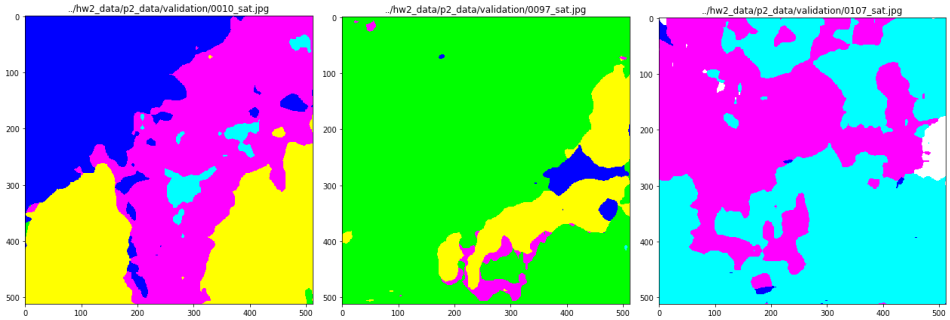
(0010_sat) (0097_sat) (0107_sat)

- 10 epoch



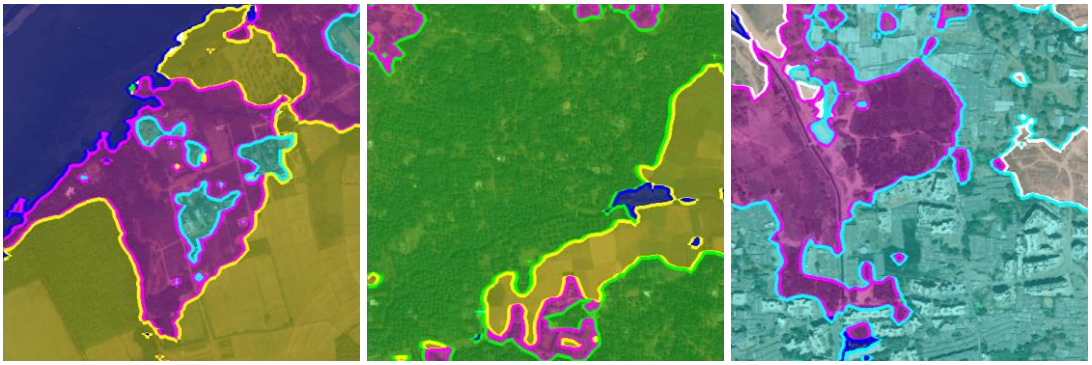
(0010_sat) (0097_sat) (0107_sat)

- 20 epoch



(0010_sat) (0097_sat) (0107_sat)

- **Result**



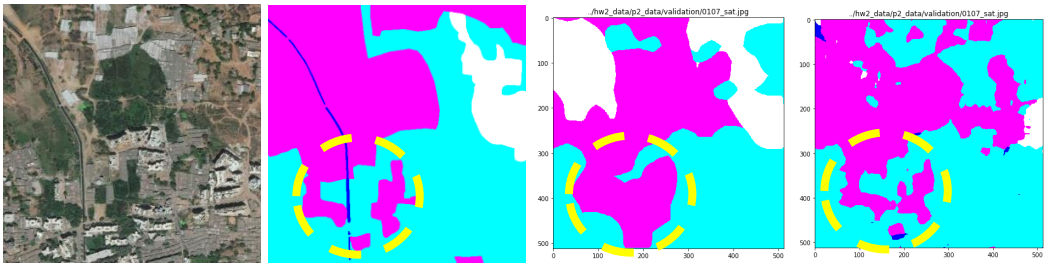
(0010_sat) (0097_sat) (0107_sat)

5. **(10%) Report mIoU 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 reasoning.**

	FCN32s (baseline)	FCN8s (improved)
Class #0	**0.73081**	**0.75480**
Class #1	**0.86538**	**0.87803**
Class #2	**0.37212**	**0.38091**
Class #3	**0.78911**	**0.80110**
Class #4	**0.61908**	**0.73942**
Class #5	**0.69110**	**0.69511**
Mean_iou	**0.677934**	**0.708230**

從下面的圖片發現，我們用 FCN8s 的圖片會看起來比較細緻，因為她有從 1/8、1/16、1/32 的 feature map 做 upsampling，然後再把這三個併起來。這樣的解析度，會比純 32 倍的 upsampling 看起來更多細節，因此在 performance 上肯定是會比較好的。

由下圖例子，我們可以看到 FCN8s 保留細節的部分比 FCN32s 好。



(orgin image) (groundtruth) (FCN32s) (FCN8s)