DLCV Homework 2

r09944003 網媒所碩一 陳竣宇

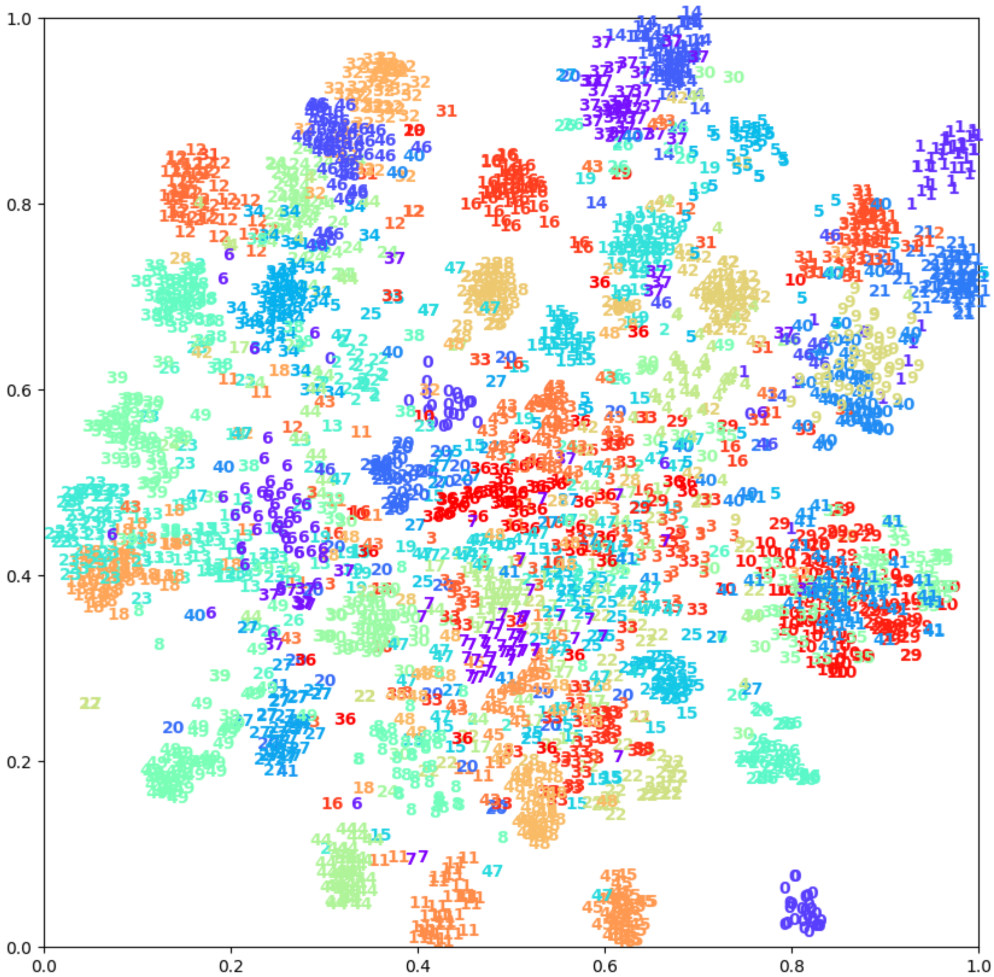
Problem 1: Image classification

1. **Network architecture**

```
Classifier(
  (cnn): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): LeakyReLU(negative_slope=0.2)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Dropout(p=0.25, inplace=False)
    (8): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): LeakyReLU(negative_slope=0.2)
    (11): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (12): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (13): LeakyReLU(negative_slope=0.2)
    (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (15): Dropout(p=0.3, inplace=False)
    (16): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (18): LeakyReLU(negative_slope=0.2)
    (19): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (21): LeakyReLU(negative_slope=0.2)
    (22): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (23): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (24): LeakyReLU(negative_slope=0.2)
    (25): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (26): Dropout(p=0.35, inplace=False)
    (27): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (28): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (29): LeakyReLU(negative_slope=0.2)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (32): LeakyReLU(negative_slope=0.2)
    (33): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (34): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (35): LeakyReLU(negative_slope=0.2)
    (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (37): Dropout(p=0.35, inplace=False)
    (38): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (39): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (40): LeakyReLU(negative_slope=0.2)
    (41): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (42): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (43): LeakyReLU(negative_slope=0.2)
    (44): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (45): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (46): LeakyReLU(negative_slope=0.2)
    (47): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (48): Dropout(p=0.35, inplace=False)
  )
  (fc): Sequential(
    (0): Linear(in_features=25088, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=1024, out_features=512, bias=True)
    (5): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): LeakyReLU(negative_slope=0.2)
    (7): Dropout(p=0.5, inplace=False)
    (8): Linear(in_features=512, out_features=50, bias=True)
  )
```

2. The accuracy on the validation set is **0.7808**

3. **TSNE visualization result**



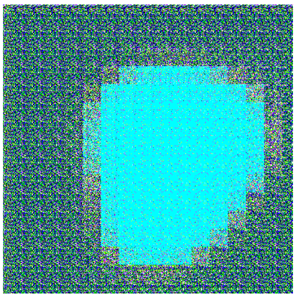
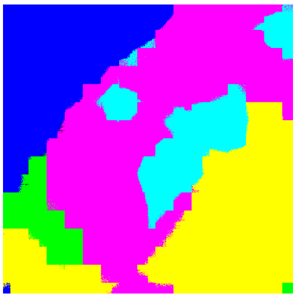
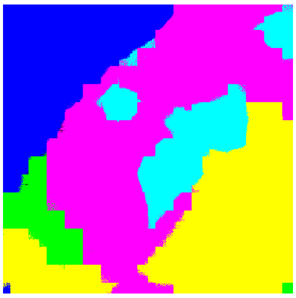
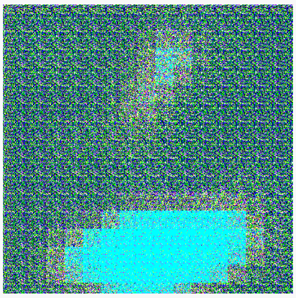
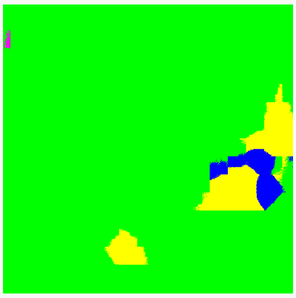
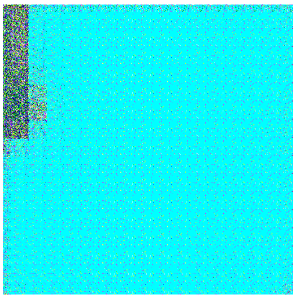
從tSNE的結果來看可以觀察到到了倒數第二層的feature已經大致有了clustering的效果，相同class圖片的feature基本上能夠落在相似的高維空間上。但是因為我本身model的accuracy並沒有非常高，因此對於一些比較容易混淆的類別，tSNE的結果就可能會有落在相似區域的情況。

Problem 2: Semantic segmentation

1. Network architecture of **VGG16-FCN32s** model

```
FCN32VGG(
  (features5): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(100, 100))
    (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=True)
    (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=True)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 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=True)
    (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=True)
    (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=True)
  )
  (fc6): Conv2d(512, 4096, kernel_size=(7, 7), stride=(1, 1))
  (bn6): BatchNorm2d(4096, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (lrelu6): LeakyReLU(negative_slope=0.2, inplace=True)
  (drop6): Dropout(p=0.5, inplace=False)
  (fc7): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
  (bn7): BatchNorm2d(4096, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (lrelu7): LeakyReLU(negative_slope=0.2, inplace=True)
  (drop7): Dropout(p=0.5, inplace=False)
  (score_fr): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
  (upscore): ConvTranspose2d(7, 7, kernel_size=(64, 64), stride=(32, 32), bias=False)
)
```

2. 在這個作業我使用1st, 20th, 40th epoch來代表我model的early, middle, final stage的結果。

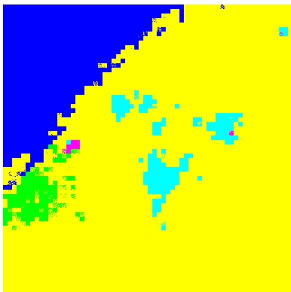
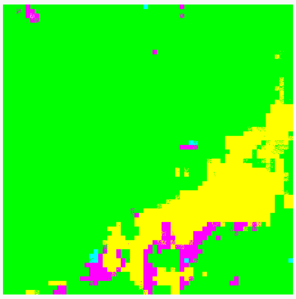
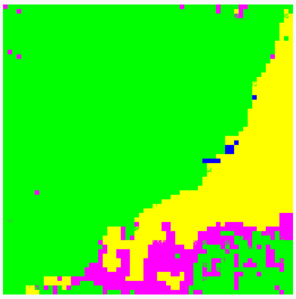
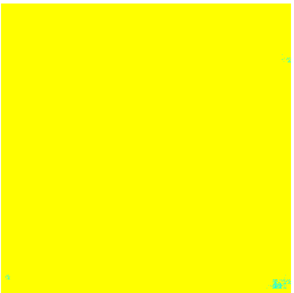
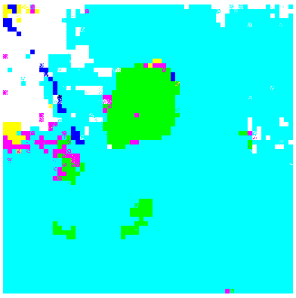
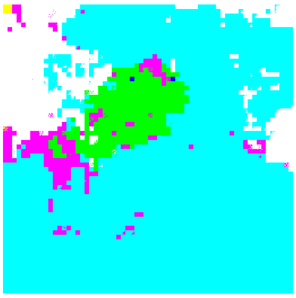
ID \ epoch	1st	20th	40th
0010			
0097			
0107			

3. Network architecture of **VGG16-FCN8s** model

```
FCN8VGG(
  (features3): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU(inplace=True)
    (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (12): ReLU(inplace=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
    (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (16): ReLU(inplace=True)
    (17): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (19): ReLU(inplace=True)
    (20): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
  )
  (features4): Sequential(
    (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
  )
  (features5): Sequential(
    (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=True)
  )
  (fc6): Conv2d(512, 4096, kernel_size=(7, 7), stride=(1, 1))
  (bn6): BatchNorm2d(4096, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (lrelu6): LeakyReLU(negative_slope=0.2, inplace=True)
  (drop6): Dropout2d(p=0.5, inplace=False)
  (fc7): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
  (bn7): BatchNorm2d(4096, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (lrelu7): LeakyReLU(negative_slope=0.2, inplace=True)
  (drop7): Dropout2d(p=0.5, inplace=False)
  (score_fr): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
  (score_pool3): Conv2d(256, 7, kernel_size=(1, 1), stride=(1, 1))
  (score_pool4_0): Conv2d(512, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn4_0): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (lrelu4_0): LeakyReLU(negative_slope=0.2, inplace=True)
  (score_pool4_1): Conv2d(1024, 2048, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn4_1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (lrelu4_1): LeakyReLU(negative_slope=0.2, inplace=True)
  (score_pool4_2): Conv2d(2048, 7, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn4_2): BatchNorm2d(7, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
(bn4_2): BatchNorm2d(7, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(lrelu4_2): LeakyReLU(negative_slope=0.2, inplace=True)
(upscore2): ConvTranspose2d(7, 7, kernel_size=(5, 5), stride=(3, 3), bias=False)
(upscore8): ConvTranspose2d(7, 7, kernel_size=(8, 8), stride=(8, 8), bias=False)
(upscore_pool4): ConvTranspose2d(7, 7, kernel_size=(2, 2), stride=(2, 2), bias=False)
```

4.

ID \ epoch	1st	20th	40th
0010			
0097			
0107			

5.

1. The mIOU of the baseline model on the validation set is **0.668957**.
2. The mIOU of the improved model on the validation set is **0.681362**.
3. 在improved model的部分我使用vgg16-8s來實作，並且經過多次嘗試得到超過baseline model mIOU的結果。
   - 對於model本身
     - 我將前面的backbone network更改為有batch normalization的VGG16
     - 從論文中可以發現，VGG16-FCN32s只用pool 5的輸出進行upsampling後產生的結果通常會是比較粗糙的，因此使用不同層級pooling後的輸出分別做upsampling，再疊加在一起，就有機會產生更好的結果。
     - 除了實作論文中的網路架構外，作者將第一個convolution layer的padding設為

    (100, 100)，但是我認為這也同時會引入不少noise，所以在improved model我將這個layer的padding定為原本VGG16的設定。
  - 原本網路架構中的crop layer是為了產生符合原圖size的output，所以只取upscore層中間的feature map。但這也會導致損失一些外圍的資訊，因此我也取消cropping並透過更改convolution / deconvolution layer的(kernel_size / stride / padding)等參數來讓輸出和原圖有相同的size。
- 在訓練的方法部分
  - 我對dataset中的training image做了額外的normalization
  - 使用focal loss替代原本的cross entropy loss，大致上可以讓結果進步0.01左右
  - 最後我在訓練後期(70 / 100)將原本的optimizer由Adam改為SGD，期望在用Adam快速收斂後SGD可以找到較optimal的點，最終結果讓我的mIOU從0.677進步為現在的0.681