

HW1report

r10922061 資工所 王竑睿

Problem 1 (10%)

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

- 本次作業 ensemble 了 torchvision 內的 vgg16_bn 和 vgg19_bn
- vgg16_bn

```
VGG(  
    (features): 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=False)  
        (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=False)  
        (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=False)  
        (24): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (25): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
        (26): ReLU(inplace=True)  
        (27): Conv2d(512, 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): ReLU(inplace=True)  
        (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): ReLU(inplace=True)  
        (33): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
        (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```

(36): ReLU(inplace=True)
(37): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(38): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(39): ReLU(inplace=True)
(40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(42): ReLU(inplace=True)
(43): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
(classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
)
)
)

```

c. vgg19_bn

```

VGG(
    (features): 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=False)
        (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=False)
        (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): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (24): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (25): ReLU(inplace=True)
        (26): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=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): ReLU(inplace=True)
    )
)

```

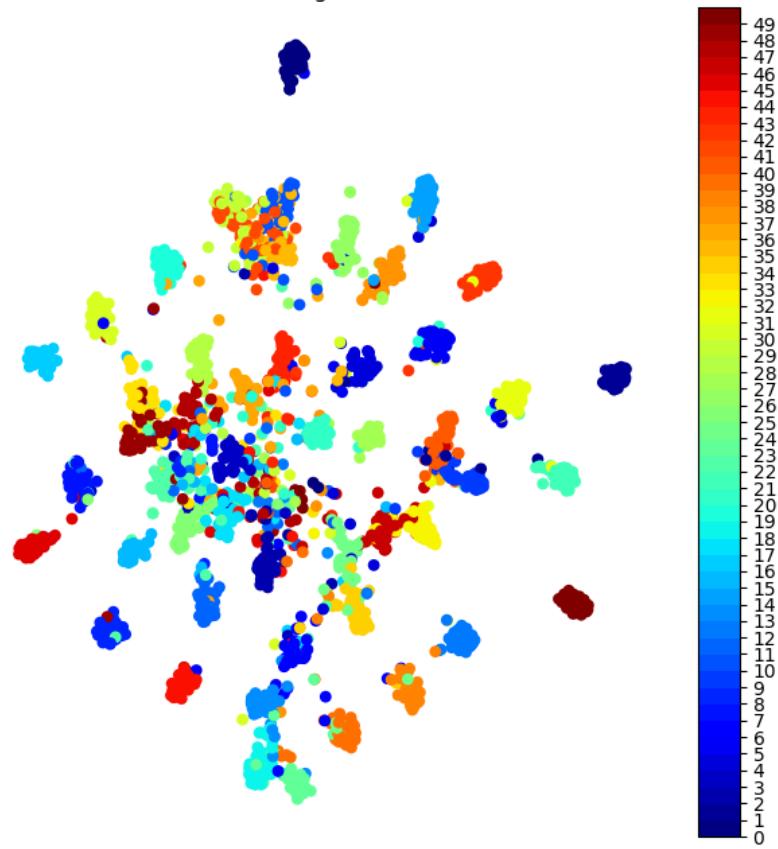
```

(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): ReLU(inplace=True)
(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): ReLU(inplace=True)
(36): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(37): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(38): ReLU(inplace=True)
(39): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(42): ReLU(inplace=True)
(43): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(44): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(45): ReLU(inplace=True)
(46): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(47): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(48): ReLU(inplace=True)
(49): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(50): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(51): ReLU(inplace=True)
(52): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
(classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
)
)
)

```

2. (2%) Report accuracy of model on the validation set. (TA will reproduce your results, error $\pm 0.5\%$)
 - a. 0.8092
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 the tSNE visualization.
 - 圖為 t-sne 將 model 倒數第二層抽出的 4096 維度向量降維到 2 維所得到的點集
 - 本次 task 共有 50 類 (0-49)。依據 colorbar 顯示的顏色，可以發現同一類的點傾向聚在相近的位置

t-sne embedding of classification



Problem 2 (20%)

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

```
SegmentationWithFCN32(  
    (backbone): 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(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=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)
)
(fcnn): Sequential(
(0): Conv2d(512, 1024, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
(1): ReLU(inplace=True)
(2): Dropout(p=0.5, inplace=False)
(3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1))
(4): ReLU(inplace=True)
(5): Dropout(p=0.5, inplace=False)
)
(score): Conv2d(1024, 7, kernel_size=(1, 1), stride=(1, 1))
(convTranspose): ConvTranspose2d(7, 7, kernel_size=(32, 32), stride=(32, 32))
)

```

2. (5%) Implement an improved model which performs better than your baseline model. Print the network architecture of this model.

a. 本次作業使用的是 ensemble VGG16-FCN32, VGG16_bn-FCN32, VGG16-FCN16

- VGG16-FCN32

```

SegmentationWithFCN32(
(backbone): 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(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=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)
)
(fcnn): Sequential(
(0): Conv2d(512, 1024, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
(1): ReLU(inplace=True)
(2): Dropout(p=0.5, inplace=False)
(3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1))
(4): ReLU(inplace=True)
(5): Dropout(p=0.5, inplace=False)
)
(score): Conv2d(1024, 7, kernel_size=(1, 1), stride=(1, 1))
(convTranspose): ConvTranspose2d(7, 7, kernel_size=(32, 32), stride=(32, 32))
)

```

- VGG16_bn-FCN32

```

SegmentationWithFCN32(
(backbone): 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=False)
(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=False)
(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=False)
(24): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(25): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(26): ReLU(inplace=True)
(27): Conv2d(512, 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): ReLU(inplace=True)
(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): ReLU(inplace=True)
(33): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(36): ReLU(inplace=True)
(37): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(38): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(39): ReLU(inplace=True)
(40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(42): ReLU(inplace=True)
(43): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(fcnn): Sequential(
(0): Conv2d(512, 1024, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
(1): ReLU(inplace=True)
(2): Dropout(p=0.5, inplace=False)
(3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1))
(4): ReLU(inplace=True)
(5): Dropout(p=0.5, inplace=False)
)
(score): Conv2d(1024, 7, kernel_size=(1, 1), stride=(1, 1))
(convTranspose): ConvTranspose2d(7, 7, kernel_size=(32, 32), stride=(32, 32))
)

```

- VGG16-FCN16

```

SegmentationWithFCN16(
(feats): 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(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)
    )
    (feat4): Sequential(
        (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (3): ReLU(inplace=True)
        (4): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (5): ReLU(inplace=True)
    )
    (feat5): Sequential(
        (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): ReLU(inplace=True)
        (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (3): ReLU(inplace=True)
        (4): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (5): ReLU(inplace=True)
    )
    (fconn): Sequential(
        (0): Conv2d(512, 1024, kernel_size=(7, 7), stride=(1, 1))
        (1): ReLU(inplace=True)
        (2): Dropout(p=0.5, inplace=False)
        (3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1))
        (4): ReLU(inplace=True)
        (5): Dropout(p=0.5, inplace=False)
    )
    (score_fconn): Conv2d(1024, 7, kernel_size=(1, 1), stride=(1, 1))
    (score_feat4): Conv2d(512, 7, kernel_size=(1, 1), stride=(1, 1))
    (upsample): Interpolate(score_fconn, score_feat4.shape[2:])
    (upsample2): Interpolate(upsample+score_feat4, x.shape[2:])
)

```

3. (5%) Report mIoU of the improved model on the validation set.

(TA will reproduce your results, error $\pm 0.5\%$)

class #0 : 0.72798
 class #1 : 0.87938
 class #2 : 0.28879
 class #3 : 0.81023
 class #4 : 0.70716
 class #5 : 0.70281

mean_iou: 0.686057

4. (5%) Show the predicted segmentation mask 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.



- 0010_sat.jpg
 - 從左到右分別為 10, 20, 30, 40, 50, 100 training epoches, ground truth



- 0097_sat.jpg
 - 從左到右分別為 10, 20, 30, 40, 50, 100 training epoches, ground truth



- 0107_sat.jpg
 - 從左到右分別為 10, 20, 30, 40, 50, 100 training epoches, ground truth

Reference

- model_p2.py, model_p2_unet.py, model_p2_fcn16.py, model_p2_fcn8.py
 1. model architecture source is highly borrowed from the blog
 2. <https://blog.csdn.net/gbz3300255/article/details/105582572>