

HW1 Report

B07901169 楊宗桓

Problem 1

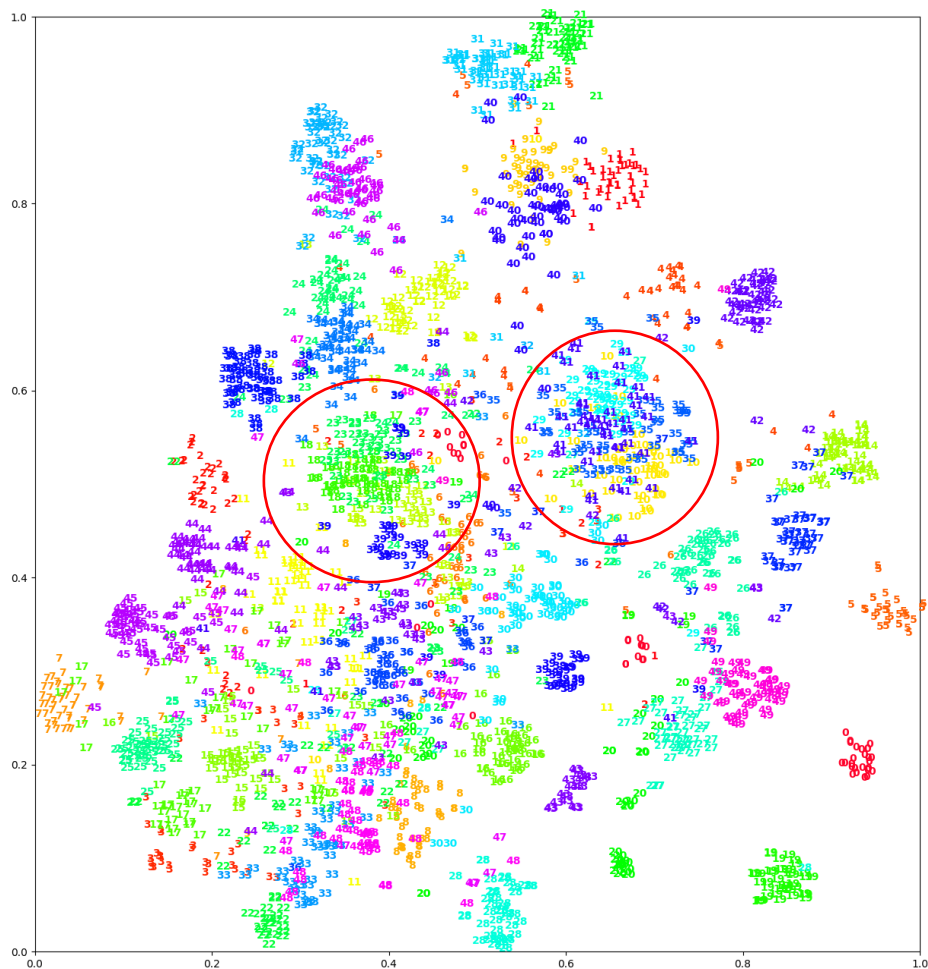
1. Print the network architecture of your model

```
vgg16_mod(
  (vgg16): 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=50, bias=True)
    )
  )
)
```

2. Report accuracy of model on the validation set:

Validation Accuracy: 82%

3. T-SNE visualization of second last layer



各類別大致上成分散，但是如圖中兩個圖所圈出來的，(13,18,23)和(10,29,35,41)
(13,18,23):



此 3 種都是植物，所以也可能對 model 較難辨別

(10,29,35,41):



由圖可看出此 4 種皆是人類，對 model 也會較難辨別

Problem 2

1. Architecture of VGG16_FCN32s

```
VGG16_FCN32(  
  (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)
)
)
(conv6): Sequential(
  (0): Conv2d(512, 4096, kernel_size=(1, 1), stride=(1, 1))
  (1): ReLU(inplace=True)
  (2): Dropout2d(p=0.5, inplace=False)
)
(conv7): Sequential(
  (0): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
  (1): ReLU(inplace=True)
  (2): Dropout2d(p=0.5, inplace=False)
)
(conv8): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
(up32): ConvTranspose2d(7, 7, kernel_size=(64, 64), stride=(32, 32), bias=False)
)

```

2. Architecture of improved model

```

VGG16_FC8(
  (vgg16): VGG(
    (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(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)
)
(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)
)
)
(conv6): Sequential(
  (0): Conv2d(512, 4096, kernel_size=(1, 1), stride=(1, 1))
  (1): ReLU(inplace=True)
  (2): Dropout2d(p=0.5, inplace=False)
)
(conv7): Sequential(
  (0): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
  (1): ReLU(inplace=True)
  (2): Dropout2d(p=0.5, inplace=False)
)
(conv8): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
(pool3): Conv2d(256, 7, kernel_size=(1, 1), stride=(1, 1))
(pool4): Conv2d(512, 7, kernel_size=(1, 1), stride=(1, 1))
(up2_1): ConvTranspose2d(7, 7, kernel_size=(4, 4), stride=(2, 2), bias=False)
(up2_2): ConvTranspose2d(7, 7, kernel_size=(4, 4), stride=(2, 2), bias=False)
(up8): ConvTranspose2d(7, 7, kernel_size=(16, 16), stride=(8, 8), bias=False)
(relu): ReLU(inplace=True)
)

```

3. mIoU of improved model

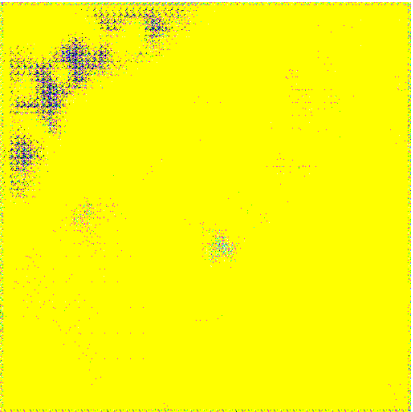
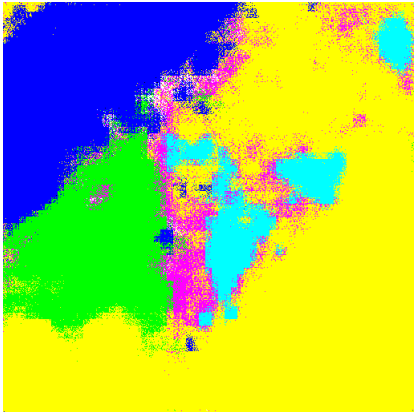
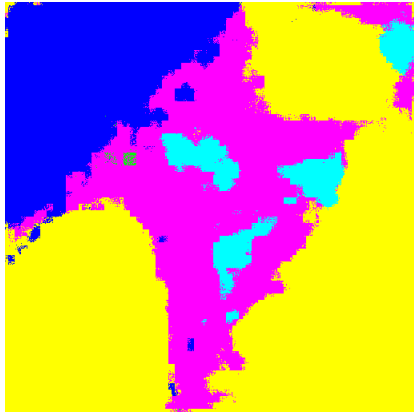
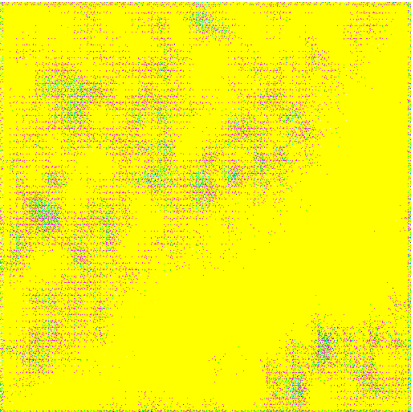

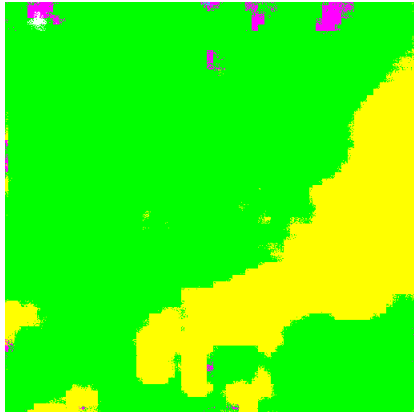
```

class #0 : 0.76535
class #1 : 0.89274
class #2 : 0.38107
class #3 : 0.80929
class #4 : 0.74225
class #5 : 0.67252

mean_iou: 0.710537

```

4.

	early	middle	final
10			
97			
107	