

Problem 1

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

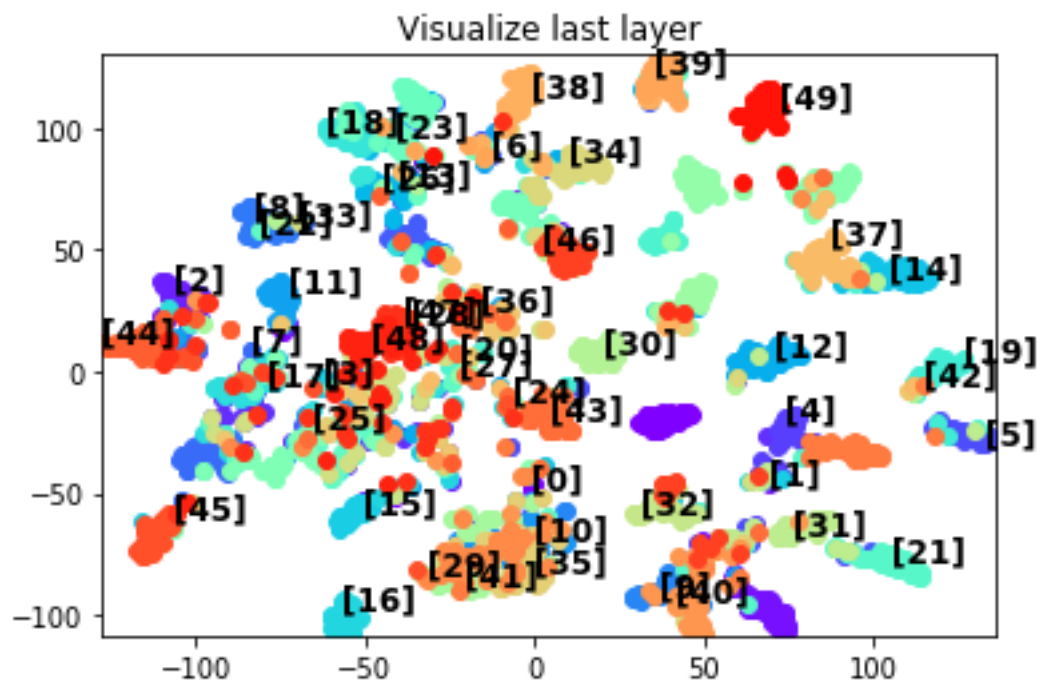
```
VGGNet(
  (features): VGG(
    (features): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(64, 64, kernel_size=(3, 3), 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(
      (fc1): Linear(in_features=25088, out_features=4096, bias=True)
      (relu1): ReLU(inplace=True)
      (dropout1): Dropout(p=0.2, inplace=False)
      (fc2): Linear(in_features=4096, out_features=4096, bias=True)
      (relu2): ReLU(inplace=True)
      (dropout2): Dropout(p=0.3, inplace=False)
      (fc5): Linear(in_features=4096, out_features=50, bias=True)
    )
  )
)
```

2. (2%) Report accuracy of model on the validation set. (TA will reproduce your results, error $\pm 0.5\%$)

```
Accuracy: 2047/2500 (81.880000%)
```

Accuracy: 81.88%

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.



透過 t-SNE 可視化後，可以看出在圖片正下方第 10、29、35、41 的類別都是重疊在一起的，表示 model 對這四類的分類效果不是很好，於是回頭去看 database 知道這四類分別為：[10]-嬰兒、[29]-成年女性、[35]-男童、[41]-女童，這其實可以發現說這個 model 其實是有學會怎麼分辨人類和其他東西的，但是對於分辨是年齡、性別的分類還是比較弱的，另外我還發現其實第 0 類別也跟上面說的這四個類別有所重疊，回去看 database 會發現第 0 類別其實應該是腳踏車，但是因為丟進去的 training data 上有不少張是人騎著腳踏車，所以這也可以合理解釋為甚麼他在分類上會跟人類這坨分類所重疊在一起。

Problem 2

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

```
fcn32s(
  (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)
  )
)

(classifier): Sequential(
  (0): Conv2d(512, 4096, kernel_size=(2, 2), stride=(1, 1))
  (1): ReLU(inplace=True)
  (2): Dropout2d(p=0.4, inplace=False)
  (3): Conv2d(4096, 1000, kernel_size=(1, 1), stride=(1, 1))
  (4): ReLU(inplace=True)
  (5): Dropout2d(p=0.5, inplace=False)
  (6): ConvTranspose2d(1000, 7, kernel_size=(64, 64), stride=(32, 32), bias=False)
)
)
```

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

```

fcn8s(
  (to_pool3): 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)
  )
  (to_pool4): 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)
  )
  (to_pool4_1): Sequential(
    (0): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (to_pool5): 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)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc6): Conv2d(512, 4096, kernel_size=(2, 2), stride=(1, 1))
  (relu1): ReLU(inplace=True)
  (drop1): Dropout2d(p=0.4, inplace=False)
  (fc7): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
  (relu2): ReLU(inplace=True)
  (drop2): Dropout2d(p=0.5, inplace=False)
  (up4x): ConvTranspose2d(4096, 256, kernel_size=(8, 8), stride=(4, 4), bias=False)
  (pool4_2xup): MaxUnpool2d(kernel_size=(2, 2), stride=(2, 2), padding=(0, 0))
  (pool4_2xup_cont): ConvTranspose2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (up8x): ConvTranspose2d(256, 7, kernel_size=(16, 16), stride=(8, 8), padding=(4, 4), bias=False)
)

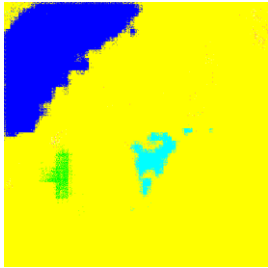
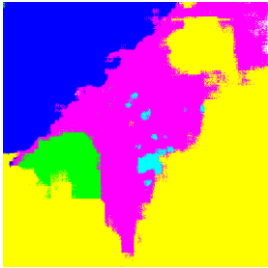
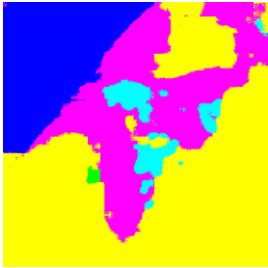
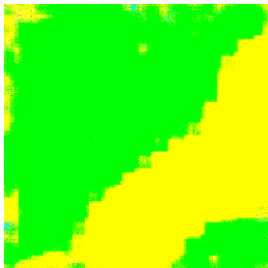
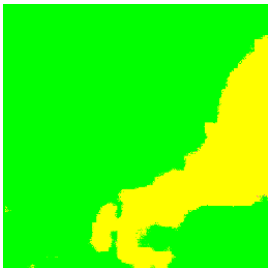
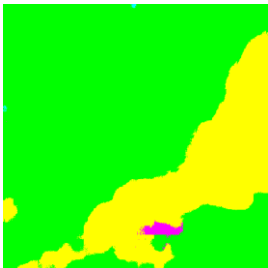
```

3. (5%) Report mIoU of the improved model on the validation set. (TA will reproduce your results, error $\pm 0.5\%$)

class #0 : 0.75637
class #1 : 0.88197
class #2 : 0.31550
class #3 : 0.81756
class #4 : 0.74336
class #5 : 0.66511
mean_iou: 0.696646

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.

Predicted segmentation on validation set using VGG-FCN8s model

Vgg16-FCN8s	Epoch 1st	Epoch 10th	Epoch 22th
0010_sat.jpg			
0097_sat.jpg			
0107_sat.jpg	