DLCV-hw1

r09946015 Allen Hung

November 02 2021

1 p1

1. Print the network architecture of your model. The figure is in Fig 1.
2. Report accuracy of model on the validation set. The val accuracy is 0.8216, the corresponding figure is in Fig 2.
3. Visualize the classification result on validation set by implementing t-SNE on output features of the second last layer. The figure is in Fig 3.

2 p2

1. Print the network architecture of your VGG16-FCN32s model. The figure is in Fig 4.
2. Print the network architecture of the improved model. ResNet50_FCN8 is in Fig 5 and Fig 6.
3. Report mIoU of the improved model on the validation set. The mIoU score is 0.698537.
4. 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's predicted masks are in Fig 8, 0097_sat.jpg's predicted masks are in Fig 9, and 0107_sat.jpg's predicted masks are in Fig 10.

3 Reference

1. github repo I've used.
   (a) https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/pix2pix/
   (b) https://github.com/NVIDIA/pix2pixHD/blob/master/models/networks.py
   (c) https://github.com/pytorch/vision/blob/5a315453da5089d66de94604ea49334a66552524/torchvision/models/resnet.py
   (d) https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

4 collaborator

1. R10944032 Bo-Yu Chen, R08922a20 Anna Huang, R09922A07, David Lou.

```
>>> model
Vgg19_bn_linear(
  (vgg19_bn): Vgg19_bn(
    (vgg_bn_pretrained): 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)
    )
  )
  (linear): Linear(in_features=25088, out_features=50, bias=True)
)
```

Figure 1: vgg19_bn_linear

```
(DLCV) allenhung@maclab-gpu-190:~/nas189/home/Courses/DLCV/hw1-allenhung1025/p1/checkpoint_vggbn_224_deepest_feature_linear_scheduler_finetune$ python3
Python 3.6.13 |Anaconda, Inc.| (default, Jun  4 2021, 14:25:59)
[GCC 7.5.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> model_10 = torch.load('./model_10.pth')
>>> model_10['val_acc']
0.8216
```

Figure 2: validation accuracy

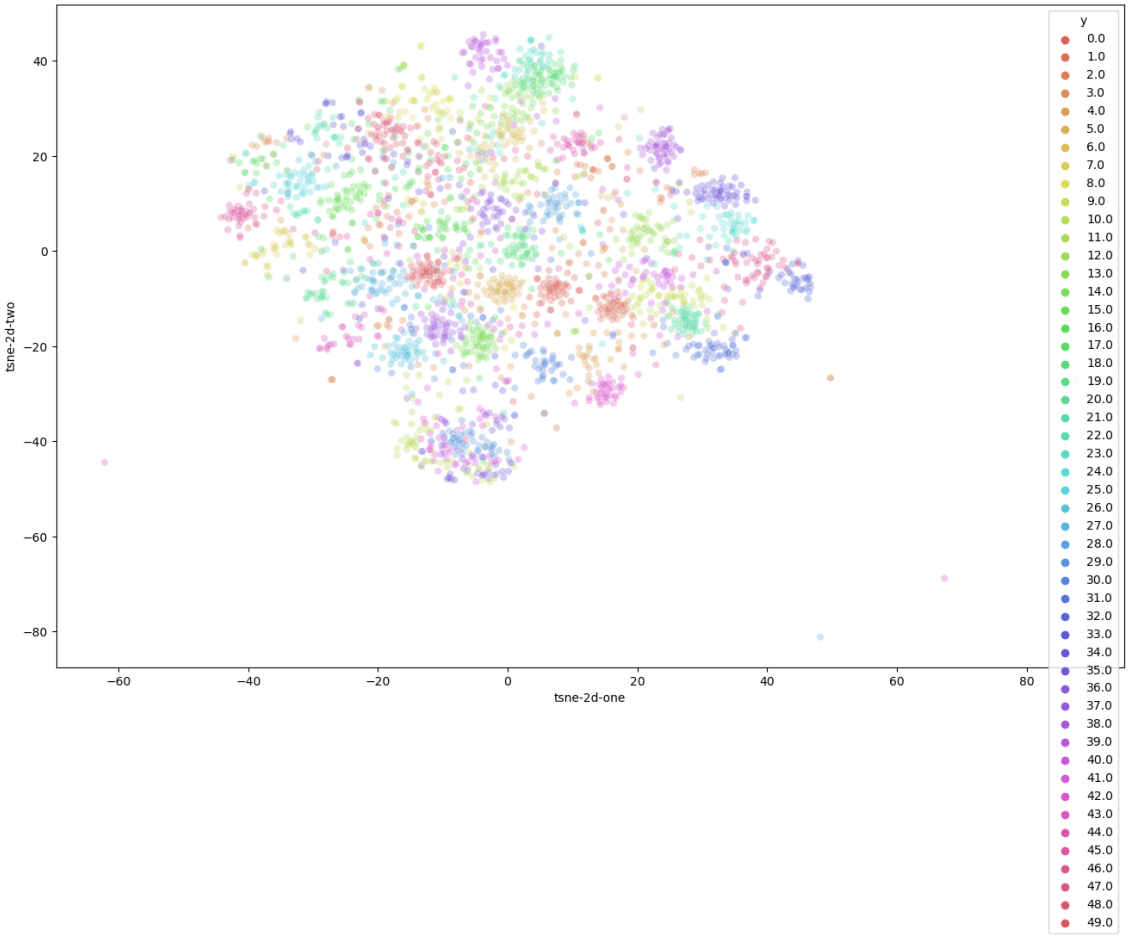

Figure 3: TSNE visulaization, there are some small clusters of the same predicted label.

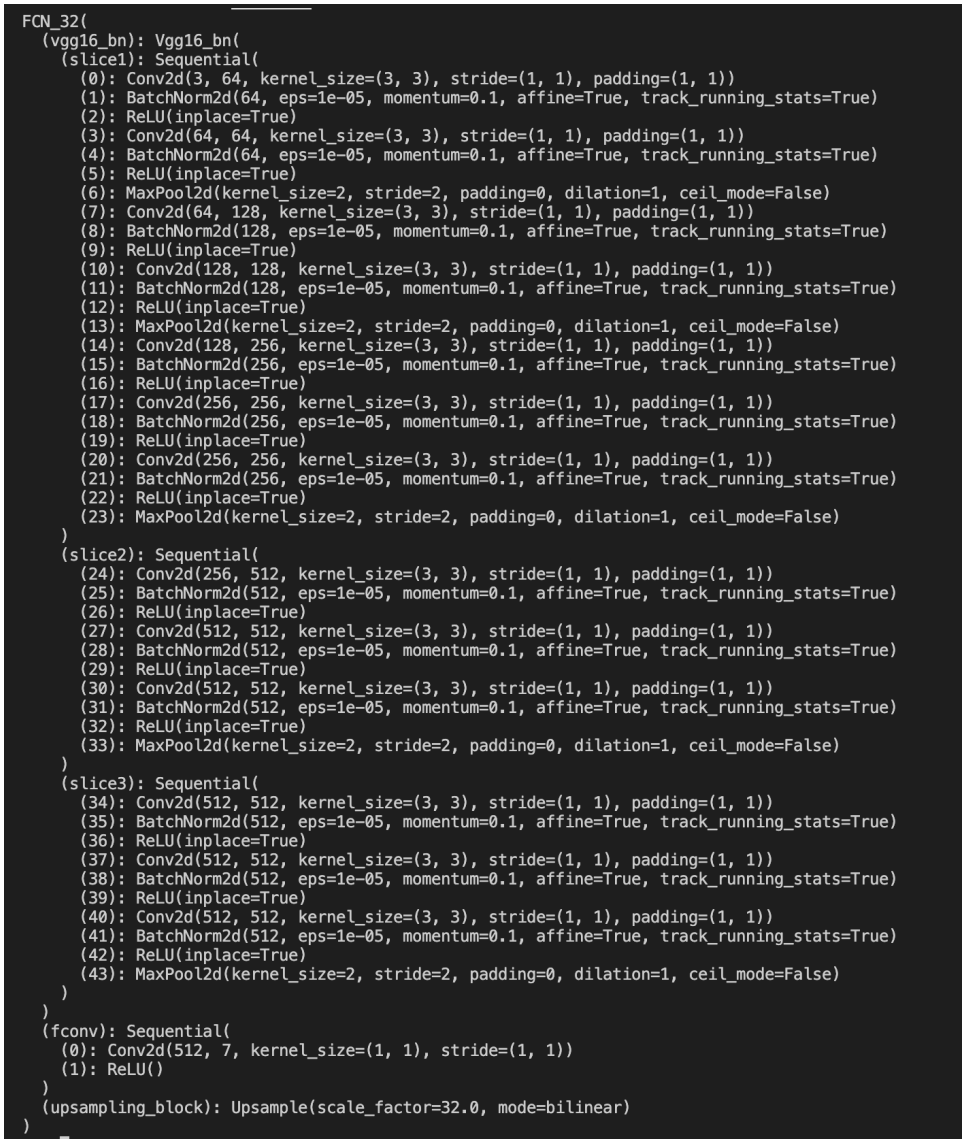

Figure 4: vgg16_fcn32

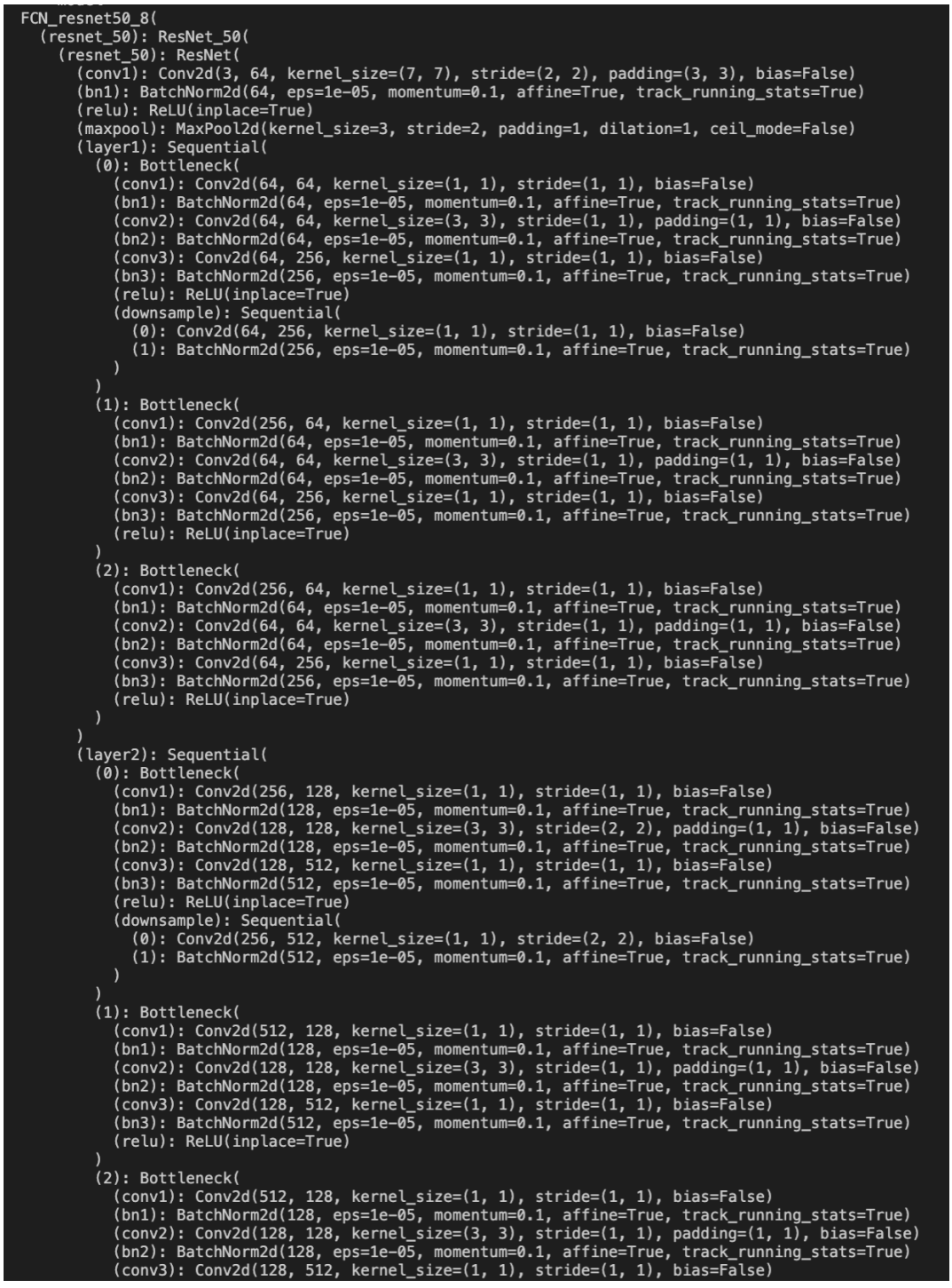

Figure 5: ResNet50_FCN8 front.

```
    (slice4): Sequential(
      (0): Sequential(
        (0): Bottleneck(
          (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
          (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (relu): ReLU(inplace=True)
          (downsample): Sequential(
            (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
            (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          )
        )
        (1): Bottleneck(
          (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (relu): ReLU(inplace=True)
        )
        (2): Bottleneck(
          (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (relu): ReLU(inplace=True)
        )
      )
    )
    (fconv0): Sequential(
      (0): Conv2d(512, 7, kernel_size=(1, 1), stride=(1, 1))
      (1): ReLU()
    )
    (fconv1): Sequential(
      (0): Conv2d(1024, 7, kernel_size=(1, 1), stride=(1, 1))
      (1): ReLU()
    )
    (fconv2): Sequential(
      (0): Conv2d(2048, 7, kernel_size=(1, 1), stride=(1, 1))
      (1): ReLU()
    )
    (upsampling_block): Upsample(scale_factor=8.0, mode=bilinear)
  )
```

Figure 6: ResNet50_FCN8 back.

```
(DLCV) allenhung@maclab-gpu-190:~/nas189/home/Courses/DLCV/hw1-allenhung1025$ python mean_iou_evaluate.py -g ./hw1_data/p2_data/validation_groundtruth/ -p output_segmentation_best/
class #0 : 0.76290
class #1 : 0.87733
class #2 : 0.33639
class #3 : 0.81315
class #4 : 0.74352
class #5 : 0.65793

mean_iou: 0.698537
```

Figure 7: validation mIoU.

Figure 8: 0010_sat.jpg predicted masks of early, middle, and final stage.

Figure 9: 0097_sat.jpg predicted masks of early, middle, and final stage.

Figure 10: 0107_sat.jpg predicted masks of early, middle, and final stage.