Problem 1

1. Network architecture

I ensemble 8 models, and use models from *torchvision.models* as feature extractors. The eight models and extra classification layers are listed as follow.

Feature extractor	Classification layer
Vgg16	Linear(512, 256) / ReLU / Linear(256, 50)
Resnet152	Linear(2048, 50)
dense161	Linear(2088, 50)
dense169	Linear(1664, 50)
resnext101_32x8d	Linear(2048, 50)
vgg19_bn	Linear(25088, 4096) / ReLU/ Dropout(0.5) / Linear(4096, 4096) / ReLU/
	Linear(4096, 50)
vgg19_bn	Linear(25088, 4096) / ReLU/ Dropout(0.5) / Linear(4096, 50)
vgg19_bn	Linear(25088, 4096) / ReLU / Dropout(0.5) / Linear(4096, 4096) / ReLU /
	Dropout(0.5) / Linear(4096, 2048) /ReLU / Dropout(0.5) / Linear(2048,
	50)

To save space, I will only print out the architecture of vgg16, dense161, and resent152.

Vgg16

```
model loaded from drive/MyDrive/DLCV/hw1/best_models_oldformat/vgg16-best.pth WG16_custom(
(backbone): Sequential(
(0): Conv2d(3, 64, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(1): ReLU(implace-True)
(2): Conv2d(34, 64, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(3): ReLU(implace-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(implace-True)
(7): Conv2d(128, 128, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(8): ReLU(implace-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(implace-True)
(12): Conv2d(256, 256, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(13): ReLU(implace-True)
(14): Conv2d(256, 256, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(15): ReLU(implace-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(implace-True)
(19): Conv2d(256, 512, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(20): ReLU(implace-True)
(21): Conv2d(128, 2512, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(20): ReLU(implace-True)
(21): Conv2d(512, 512, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(22): ReLU(implace-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(implace-True)
(28): Conv2d(512, 512, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(27): ReLU(implace-True)
(28): Conv2d(512, 512, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(27): ReLU(implace-True)
(28): Conv2d(512, 512, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(29): ReLU(implace-True)
(20): Conv2d(512, 512, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1))
(29): ReLU(implace-True)
(20): Conv2d(512, 512, kernel_size-(3,
```

Figure 1: model architecture of vgg16

Dense161

```
DenseNet
    niseNet(
(features): Sequential(
(conv0): Conv2d(3, 96, kernel_size-(7, 7), stride-(2, 2), padding-(3, 3), bias-False)
(norm0): EactNorm2d(96, eps-le-05, momentum-0.1, affine-True, track_running_stats-True)
(relu0): ReLU(implace-True)
                             MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
          (denseblock1): _DenseBlock(
(denselayer1): _DenseLaye
                                     rl): _DenseLayer(
BatchNorm2d(96, eps=1e=05, momentum=0.1, affine=True, track_running_stats=True)
                   (norm1): BatchNorm2d(96, ep
(relu1): ReLU(inplace=True)
                   vresur: MeLU(inplace=True)
(conv1): Conv2(86, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
(norm2): BatchNorm2d(192, eps=1e=05, momentum=0.1, affine=True, track_running_stats=True)
                   (relu2): ReLU(inplace=True)
(conv2): Conv2d(192, 48, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
              /
(denselayer2): _DenseLayer(
  (norm1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                   (norm1): BatchNorm2d(144, eps-1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu1): RatU(implace=True)
(conv1): Conv2d(144, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
(norm2): BatchNorm2d(192, eps-1e-05, momentum=0.1, affine=True, track_running_stats=True)
(-1.0.0).net_in=10.00=True)
                   (relu2): ReLU(inplace=True)
(conv2): Conv2d(192, 48, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
              / (denselayer3): _DenseLayer(
  (norm1): BetchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                   (norm1): BatchNorm2d(182, eps-1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu1): ReLU(inplace=True)
(conv1): Conv2d(182, 182, kernel_size=(1, 1), stride=(1, 1), bias=False)
(norm2): BatchNorm2d(182, eps-1e-05, momentum=0.1, affine=True, track_running_stats=True)
(-1.-0.) = norm2d(182, eps-1e-05, momentum=0.1, affine=True, track_running_stats=True)
                   (relu2): ReLU(inplace=True)
(conv2): Conv2d(192, 48, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
             )
(denselayer4): _Denselayer(
  (norm1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu1): ReLU(implace=True)
(conv1): Conv2d(240, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (norm2): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu2): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                   (relu2): ReLU(inplace=True)
(conv2): Conv2d(192, 48, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
                   ense.ayero/: _Dense.ayer(
(norm1): Bat.Dhorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relul): ReLU(implace=True)
(conv1): Conv2d(288, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
(norm2): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                   (relu2): ReLU(inplace=True)
(conv2): Conv2d(192, 48, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
                  (denselayer6): _DenseLayer(
(norm1): BatchNorm2d(336, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
                      (norm1): BatchNorm2d(336, eps-1e-UD, momentum=U.1, arrine=irue, trans_ummine_state irue* (relul): ReU(implace=True) (corul): Conv2d(336, 192, kernel_size=(1, 1), stride=(1, 1), bias=False) (norm2): BatchNorm2d(192, eps-1e-UD, momentum=U.1, affine=True, track_running_state=True) (relul2): ReU(implace=True) (conv2): Conv2d(192, 48, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
             )
(transition1): _Transition(
(norm): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReUU(implace=True)
(conv): Convd(384, 198, kernel_size=(1, 1), stride=(1, 1), bias=False)
(pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
                  sensellock2: _uensesiock
(denselsyer): _DenseLayer (eps-le-05, momentum-0.1, sffine-True, track_running_stats=True)
(norul): BatchNorm2d(192, eps-le-05, momentum-0.1, sffine-True, track_running_stats=True)
(conv1): Conv2d(192, 192, kernel_size-(1, 1), stride-(1, 1), bias-False)
(norul): BatchNorm2d(192, eps-le-05, momentum-0.1, sffine-True, track_running_stats=True)
(relu2): BatClin(Inplace-True)
(conv2): Conv2d(192, 48, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1), bias-False)
                  (denselaver3): DenseLaver(
                       emselsyer3): Demselsyer(
(norm1): BatchNorm2d(288, eps-1e-05, momentum-0.1, affine-True, track_running_stats=True)
(relu1): ReLU(implace-True)
(conv1): Conv2d(288, 192, kernel_size-(1, 1), stride-(1, 1), bias=False)
(norm2): BatchNorm2d(192, eps-1e-05, momentum-0.1, affine-True, track_running_stats=True)
(conv2): Conv2d(192, 48, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1), bias=False)
                  )
(denselayer4): _DenseLayer(
  (norm1): BatchNorm2d(336, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relun): ReLU(inplace=True)
  (convu): Corne2(336, 182, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (norm2): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                       (relu2): ReLU(inplace=True)
(conv2): Conv2d(192, 48, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
```

Figure 2: model architecture of dense161

Resnet152

```
ResNet(
(conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
(bn1): BatchNorm2d(66, eps=4e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu): Reu(unplace=True)
(maxpool): NaxPool2d(kernel_size=0, stride=2, padding=1, dilation=1, ceil_mode=False)
(layer1): Sequential(
(e): Bottleneck(
(conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
(conv): Conv2d(64, 64, kernel_size=(2, 3), stride=(1, 1), padding=(1, 1), bias=False)
(conv): Conv2d(64, 64, kernel_size=(2, 3), stride=(1, 1), padding=(1, 1), bias=False)
(conv): Conv2d(64, 556, kernel_size=(1, 1), stride=(1, 1), bias=False)
(conv): Reu(unplace=True)
(domorangle): Sequential(
(e): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
(domorangle): Sequential(
(e): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
(1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(1): RatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  (4): Bottleneck(
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             4): Bottleneck(
(conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              (5): Bottleneck(
(5): Bottleneck(
(comv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
(conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
(conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padiang=(1, 1), bias=False)
(bn2): BatchNorm2d(128, eps=1e=05, momentum=0.1, affine=True, track_running_stats=True)
(conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn3): BatchNorm2d(512, eps=1e=05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
                            ) (1): Bottleneck(
(conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=false)
(bn1): Batchorm2d(64, eps=1e=08, momentum=0.1, affine=True, track_running_stats=True)
(conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=false)
(bn2): Batchorm2d(64, eps=1e=08, momentum=0.1, affine=True, track_running_stats=True)
(conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=false)
(bn3): Batchorm2d(256, eps=1e=08, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              (G): Bottleneck(
(Conv1): Conv2d(S12, 128, Kernel_size-(1, 1), stride=(1, 1), bias=False)
(bnl): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(Conv2): Conv2d(128, 128, Kernel_size-(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(128, ps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(Conv3): Conv2d(128, 512, Kernel_size-(1, 1), stride=(1, 1), bias=False)
(bn3): BatchNorm2d(S12, ps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
                            )
(2): Bottleneck(
(com/1): Com/2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn1): Batchomrad(64, eps=le=05, momentum=0.1, affine=True, track_running_stats=True)
(com/2): Com/2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): Batchomrad(64, eps=le=05, momentum=0.1, affine=True, track_running_stats=True)
(com/3): Com/2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn3): Batchomrad(256, eps=le=05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  (7): Bottleneck(
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             7): Bottleneck(
(conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn1): BatchNorm2d(128, eps=le-05, momentum=0.1, affine=True, track_running_stats=True)
(conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(128, eps=le-05, momentum=0.1, affine=True, track_running_stats=True)
(conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn3): BatchNorm2d(512, eps=le-05, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
             (reary). Returning.accenture.)

(a): Bottleneck(
(cord)): Cornol(265, 128, kernel_size-(1, 1), stride-(1, 1), biss-false)
(cord): Cornol(265, 128, kernel_size-(1, 1), stride-(1, 1), biss-false)
(cord): Cornol(2618, 128, kernel_size-(2, 3), stride-(2, 2), padding(1, 3), biss-false)
(bn2): Batchbornol(128, epsal-e95, momentum-0.1, affine-True, track_running_stats-True)
(cord): Cornol(218, 128, kernel_size-(1, 1), stride-(1, 1), biss-false)
(bn3): Batchbornol(512, epsal-e95, momentum-0.1, affine-True, track_running_stats-True)
(relu): RetU(inplace-True)
(downsample): Sequential(
(0): Cornol(2636, 128, kernel_size-(1, 1), stride-(2, 2), biss-false)
(1): Batchbornol(512, epsal-e-05, momentum-0.1, affine-True, track_running_stats-True)
)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  (layer4): Sequential(
(0): Bottleneck(
(conv1): Conv2d(1824, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn1): BatthNorm2d(512, eps=1e-86, 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): BatthNorm2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
(conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
(pa): BatthNorm2d(2048, eps=1e-86, momentum=0.1, affine=True, track_running_stats=True)
(dowssmaple): Sequential(
(0): Conv2d(1824, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
(1): BatthNorm2d(2048, eps=1e-85, momentum=0.1, affine=True, track_running_stats=True)
                                                      (): Bottleneck(
(comu1): Comzdd(s12, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn): Batthlorm2d(128, eps=le=08, momentum=0.1, affine=True, track_running_stats=True)
(comu2): Comuzd(128, 218, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn): Batthlorm2d(128, eps=le=08, momentum=1, affine=True, track_running_stats=True)
(comu2): Comuzd(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn): Batthlorm2d(512, eps=le=08, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (3): Bottleneck(
((com/1): conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=false)
(bn1): Battlonm2d(512, eps=1e=05, momentum=0.1, affine=True, track_running_stats=True)
((com/2): conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=false)
(bn2): Battlonm2d(512, pas=1e=05, momentum=0.1, affine=True, track_running_stats=True)
((com/3): conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=false)
(bn3): Battlonm2d(2048, pcs=1e=05, momentum=0.1, affine=True, track_running_stats=True)
((relu): ReLU(Inplace=True)
                                         )
(2): Bottleneck(
(conv1): Conv3d(512, 128, kernel_size-(1, 1), stride-(1, 1), bias=False)
(bn1): Batchkomm2d(128, eps=1e-08, nomentum=0.1, affine=True, track_running_stats=True)
(conv2): Conv3d(128, 128, kernel_size-(3, 3), stride-(1, 1), padding-(1, 1), bias=False)
(bn2): Batchkomm2d(128, eps=1e-08, nomentum=0.1, affine=True, track_running_stats=True)
(conv3): Conv3d(128, 512, kernel_size-(1, 1), stride-(1, 1), bias=False)
(bn3): Batchkomm2d(512, eps=1e-05, nomentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    (2): Bottleneck(
(com/1): ComV2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
(com/1): ComV2d(2048, 512, kernel_size=(3, 1), affine=True, track_running_stats=True)
(com/2): ComV2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(512, eps=1e-08, momentum=0.1, affine=True, track_running_stats=True)
(com/3): ComV2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn3): BatchNorm2d(2048, eps=1e-08, momentum=0.1, affine=True, track_running_stats=True)
(relu): ReLU(inplace=True)
                                           (3): Bottleneck(
                                                        ): Bottleneck(
(conv1): Convoid(si2, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
(bn1): Batchkornad(128, eps=1e-08, momentum=0.1, affine=frue, track_running_stats=frue)
(conv2): Convoid(28, 128, kernel_size=(3, 3), stride=(1, 1), podding=(1, 1), bias=False)
(bn2): Batchkornad(128, eps=1e-08, momentum=0.1, affine=frue, track_running_stats=frue)
(cons): Convoid(128, sizk, kernel_size=(1, 1); stride=(1, 1), bias=False)
(bn3): Batchkornad(siz, eps=1e-08, momentum=0.1, affine=frue, track_running_stats=frue)
(rd1): Reul(inglace=frue)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=2048, out_features=50, bias=True)
```

Figure 3: model architecture of resnet152

2. Validation accuracy = 0.8272

3. Visualize result

Choose vgg16 + classification layers. Extract the output of the second last layer, which is fc.0. Use sklearn.manifold.TSNE to perform TSNE, and transform 256-dim vectors into 2-dim vectors..

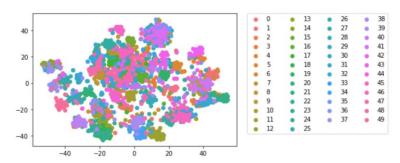
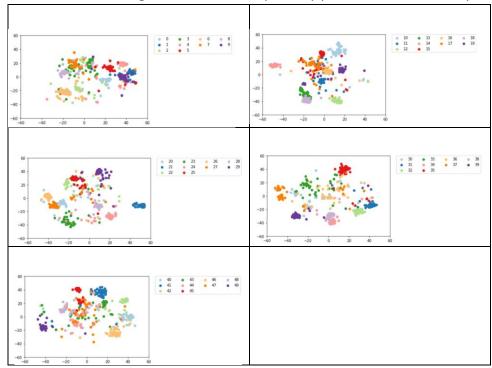


Figure 4: TSNE result

Discussion:

Because it's hard to distinguish 50 colors, I separately plot each 10 class in a plot.



Most class data clustered into one or several clusters, however some classes of data are still are still scattered. For example, class 30, 33, 47 are relatively scattered, which are respectively mushrooms, mouse, and otter. In addition, and different class clusters aren't totally separated. For example, class 18 and 13 overlap each other, which are two kinds of trees. Apparently, these two classes are very similar, so their projected vectors are similar, too.

Problem 2

1. Model Architecture

```
fcn32_lb(
    (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)
    (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)
    (20): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): ReLU(inpl
```

Figure 5: model architecture of FCN32s

Add three convolution layers after vgg16 feature extractor, and up sample the feature map by 32 times by a single convolution transpose layer.

2. Improved Model Architecture
Implement FCN8s proposed in [1]. The model architecture is as follow:

```
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)
(16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(15): ReLU(inplace=True)
(16): MaxPool2d(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)
)
(5): 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))
(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)
)
(5): ReLU(inplace=True)
(6): MaxPool2d(kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(7): ReLU(inplace=True)
(8): ReLU(inplace=True)
(9): ReLU(inplace=True)
(10): ReLU(inplace=True)
(10
```

Figure 6: model architecture of FCN8s

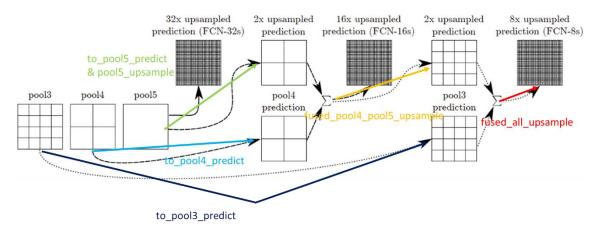
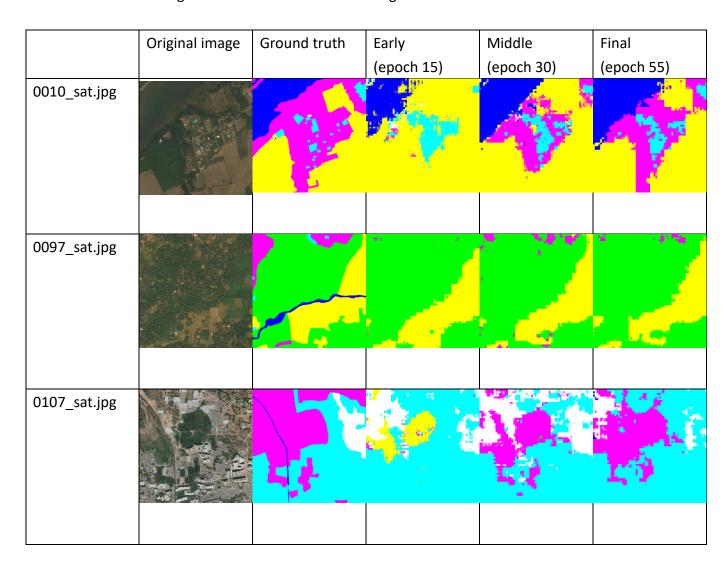


Figure 7: corresponding layers in the original paper [1][2]

- 3. Validation mIoU = 0.725185
- 4. Predicted segmentation mask of different stages



Reference

- [1] "Fully Convolutional Networks for Semantic Segmentation", Evan Shelhamer, Jonathan Long, and Trevor Darrell, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 39, NO. 4, APRIL 2017
- [2] https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1