DLCV HW2 Report

Shuo-En Chang
R08922A02

November 4, 2020

1 Image classification

1.1 Print the network architecture of your model.

```
resnext101(
  (backbone): ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (layer1): Sequential(
      (0): Bottleneck(
        (conv1): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
            groups=32, bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
        (conv3): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
        (relu): ReLU(inplace=True)
        (downsample): Sequential(
          (0): 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): Bottleneck(
        (conv1): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
            groups=32, bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
        (conv3): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
```

```
    (relu): ReLU(inplace=True)
  )
  (2): Bottleneck(
    (conv1): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        groups=32, bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
)
(layer2): Sequential(
  (0): Bottleneck(
    (conv1): Conv2d(256, 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),
        groups=32, bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(512, 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)
    (downsample): Sequential(
      (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
    )
  )
  (1): Bottleneck(
    (conv1): Conv2d(512, 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),
        groups=32, bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(512, 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)
  )
  (2): Bottleneck(
    (conv1): Conv2d(512, 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),
        groups=32, bias=False)
```

```
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(512, 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)
  )
  (3): Bottleneck(
    (conv1): Conv2d(512, 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),
        groups=32, bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(512, 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)
  )
)
(layer3): Sequential(
  (0): Bottleneck(
    (conv1): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
        groups=32, bias=False)
    (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (relu): ReLU(inplace=True)
    (downsample): Sequential(
      (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
    )
  )
  (1): Bottleneck(
    (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        groups=32, bias=False)
    (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (2): Bottleneck(
    (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

```
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(3): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(4): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(5): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(6): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
```

```
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(7): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(8): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(9): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(10): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
```

```
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(11): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(12): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(13): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(14): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

```
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(15): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(16): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(17): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
)
(18): Bottleneck(
  (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      groups=32, bias=False)
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (relu): ReLU(inplace=True)
```

```
  )
  (19): Bottleneck(
    (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        groups=32, bias=False)
    (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (20): Bottleneck(
    (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        groups=32, bias=False)
    (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (21): Bottleneck(
    (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        groups=32, bias=False)
    (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (22): Bottleneck(
    (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        groups=32, bias=False)
    (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (conv3): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
)
(layer4): Sequential(
```

```
    (0): Bottleneck(
      (conv1): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (conv2): Conv2d(2048, 2048, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
          groups=32, bias=False)
      (bn2): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (conv3): Conv2d(2048, 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, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (conv2): Conv2d(2048, 2048, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
          groups=32, bias=False)
      (bn2): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (conv3): Conv2d(2048, 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, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (conv2): Conv2d(2048, 2048, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
          groups=32, bias=False)
      (bn2): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (conv3): Conv2d(2048, 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)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=2048, out_features=50, bias=True)
)
```

1.2 Report accuracy of model on the validation set.

Accuracy of model on the validation set is 0.878.

1.3 Visualize the classification result on validation set by implementing t-SNE on output features of the second last layer.

We can see the output features of the second last layer on the validation set in Figure 1. In fact, we can see that most classes can be gathered into different groups. I think it can be said that the output features of the second last layer can indeed be used for classification. Some of them do not cluster well (e.g. upper left part of the image), maybe they will be learnt in the last layer. Otherwise, this is the reason why the model cannot achieve 100% accuracy.

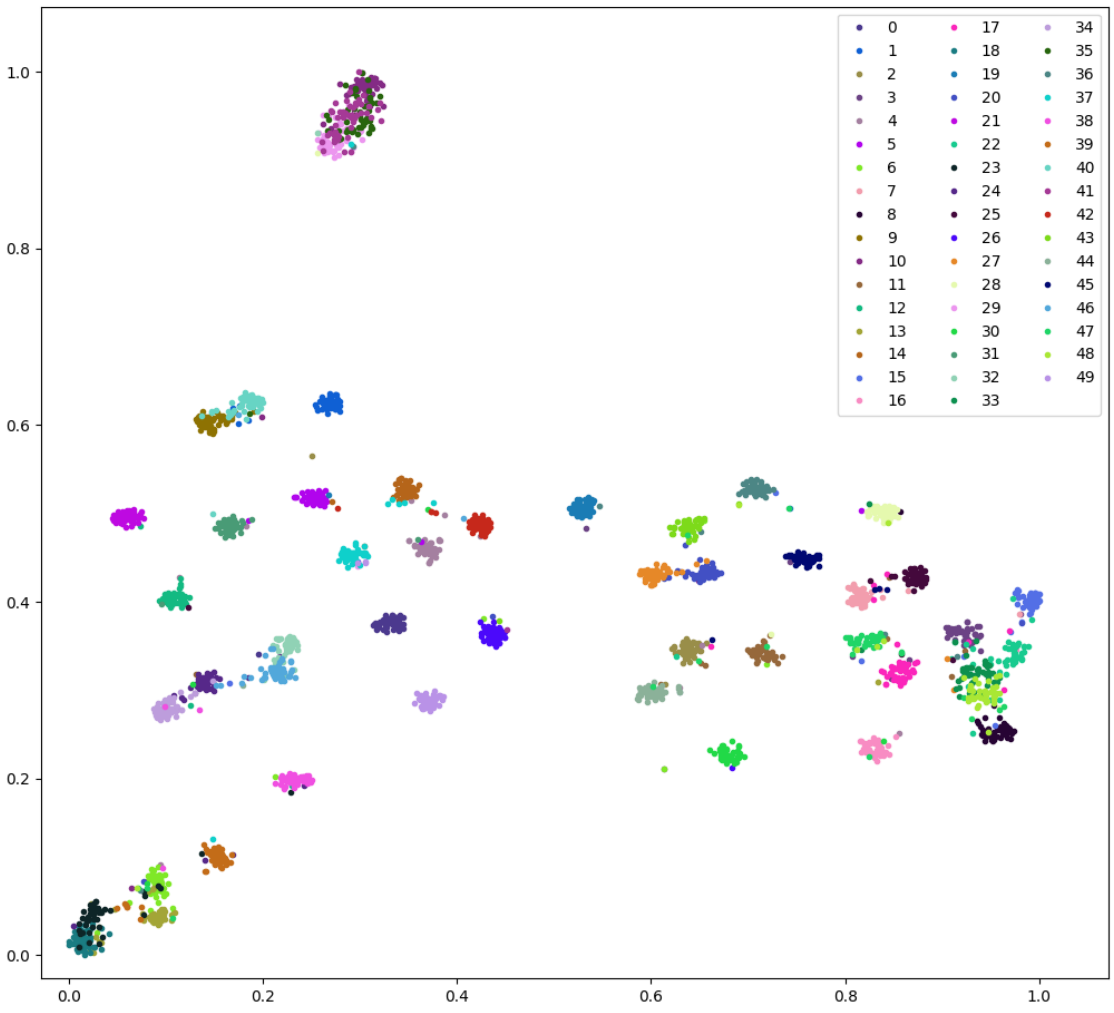


Figure 1: Features of the second last layer on validation set

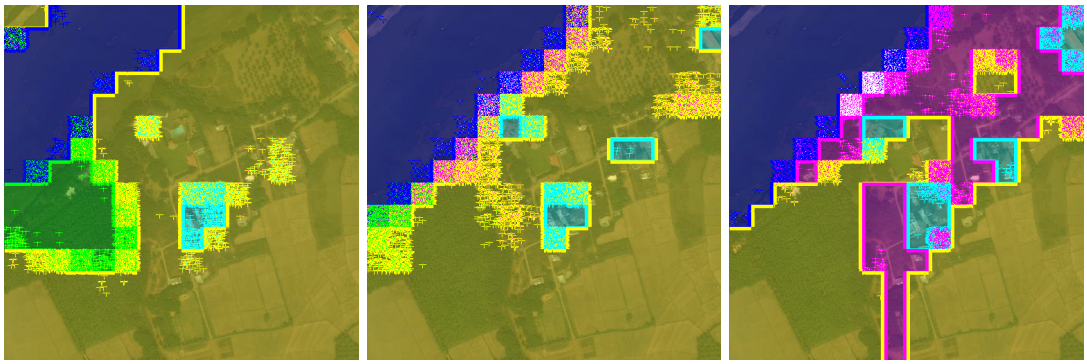
2 Semantic segmentation

2.1 Print the network architecture of your VGG16-FCN32s model.

```
VGGFCN32s(
  (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)
  )
  (fcn): Sequential(
    (0): Conv2d(512, 4096, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Dropout2d(p=0.1, inplace=False)
    (3): Conv2d(4096, 4096, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): Dropout2d(p=0.1, inplace=False)
  )
  (classifier): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
  (upsampled): ConvTranspose2d(7, 7, kernel_size=(32, 32), stride=(32, 32))
)
```

2.2 Show the predicted segmentation mask during the training stage.

Please refer to Figures 2 to 4

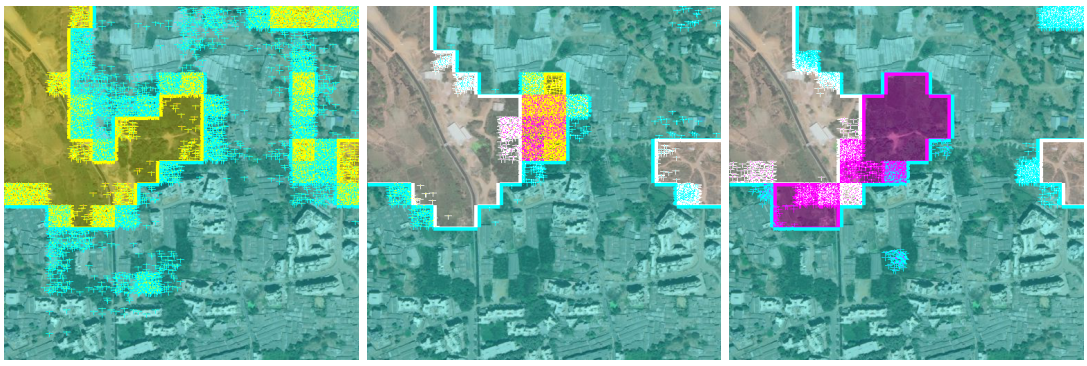


(a) Epoch=1 (b) Epoch=5 (c) Epoch=9

Figure 2: predicted segmentation mask for 0010_sat.jpg

(a) Epoch=1 (b) Epoch=5 (c) Epoch=9

Figure 3: predicted segmentation mask for 0097_sat.jpg



(a) Epoch=1 (b) Epoch=5 (c) Epoch=9

Figure 4: predicted segmentation mask for 0107_sat.jpg

2.3 Implement an improved model.

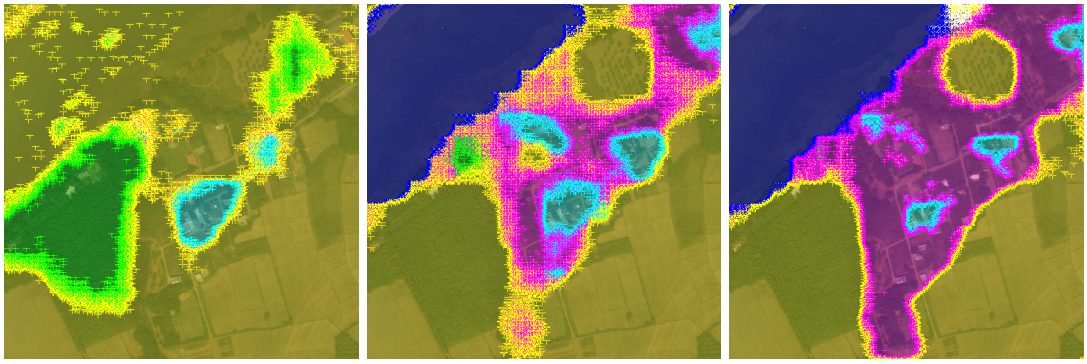
```
EnSegNet8(
  (block1): Sequential(
```

```
  (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)
)
(block2): Sequential(
  (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)
)
(block3): Sequential(
  (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)
)
(block4): Sequential(
  (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)
)
(block5): Sequential(
  (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)
)
(deconv1): Sequential(
  (0): ConvTranspose2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
      output_padding=(1, 1))
  (1): ReLU(inplace=True)
)
(deconv2): Sequential(
  (0): ConvTranspose2d(512, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
      output_padding=(1, 1))
  (1): ReLU(inplace=True)
)
(deconv3): Sequential(
  (0): ConvTranspose2d(256, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
      output_padding=(1, 1))
  (1): ReLU(inplace=True)
)
```

```
  (deconv4): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
        output_padding=(1, 1))
    (1): ReLU(inplace=True)
  )
  (deconv5): Sequential(
    (0): ConvTranspose2d(64, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
        output_padding=(1, 1))
    (1): ReLU(inplace=True)
  )
  (classifier): Conv2d(32, 7, kernel_size=(1, 1), stride=(1, 1))
)
```

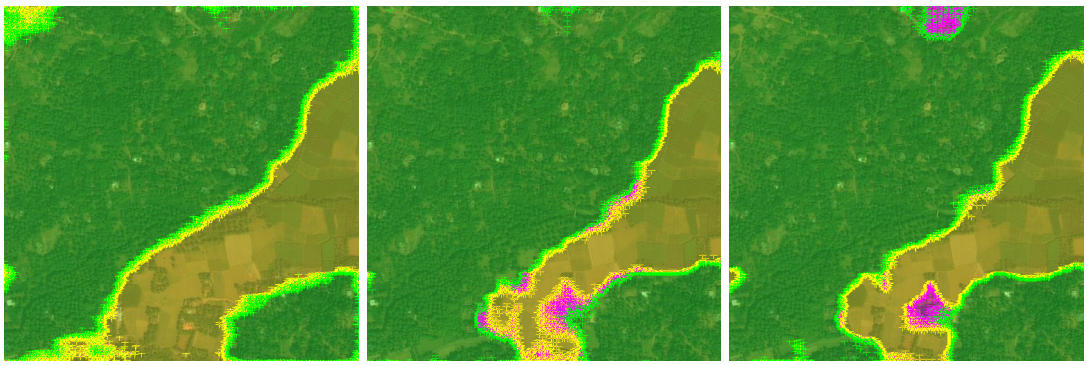
2.4 Show the predicted segmentation mask during the training stage.

Please refer to Figures 5 to 7



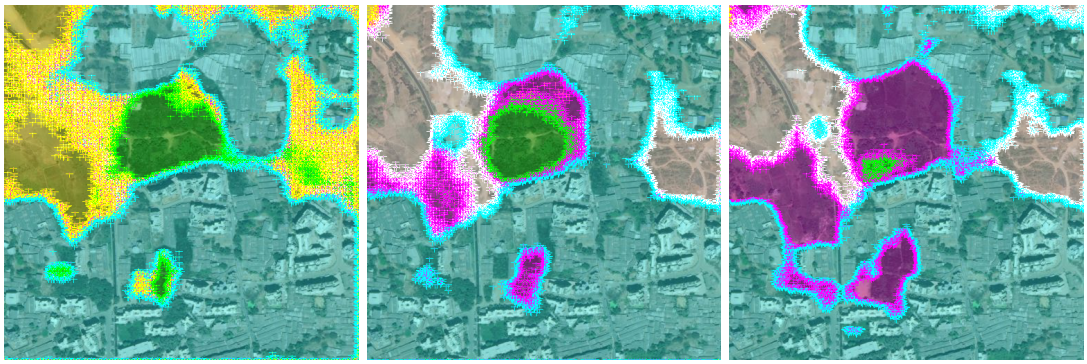
(a) Epoch=1 (b) Epoch=11 (c) Epoch=23

Figure 5: predicted segmentation mask for 0010_sat.jpg (improved)



(a) Epoch=1 (b) Epoch=11 (c) Epoch=23

Figure 6: predicted segmentation mask for 0097_sat.jpg (improved)



(a) Epoch=1 (b) Epoch=11 (c) Epoch=23

Figure 7: predicted segmentation mask for 0107_sat.jpg (improved)

2.5 Report mIoU score of both models on the validation set.

	VGGFCN32s	VGGFCN8s	My improved model
mIoU	0.689	0.695	**0.700**

My improvement is to try to solve the "square" problem (i.e. not a smooth curve on the boundary) we encountered in figs. 2 to 4. I think this is because, in the FCN[2] model, it will upsample the size of features 32 times at once.

Therefore, I first add some additional transposed convolutional layers and only upsample the model twice at a time. It is similar to Segnet[1], but since I did not add any other convolutional layer between the two transposed convolutional layers, it is not exactly the same. I found that after the modification, the mIoU of the model on the validation set can be increased to 0.691.

After that, I do the experience with FCN-8s[2] architecture. Compared with FCN-32s[2], it can get a significant improvement on the validation set. Inspired by this, I added a shortcut to the output of pool3 and pool4 in my improved model and finally improved the performance to 0.700.

The reason for this improvement can be found in figs. 5 to 7, the "square" problem has been solved. I think this is because the model can have many opportunities to learn how to upsampling without having to do it all at once.

3 Collaboration

I had discussed this assignment with Tzu-Hsuan Weng(R08922144), Jie Ting(R08922130), Austin Tsai(R08922086).

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[2] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.