

DLCV HW 1

B08902134, 曾揚哲

1 Image Classification

1.1 Model Structure

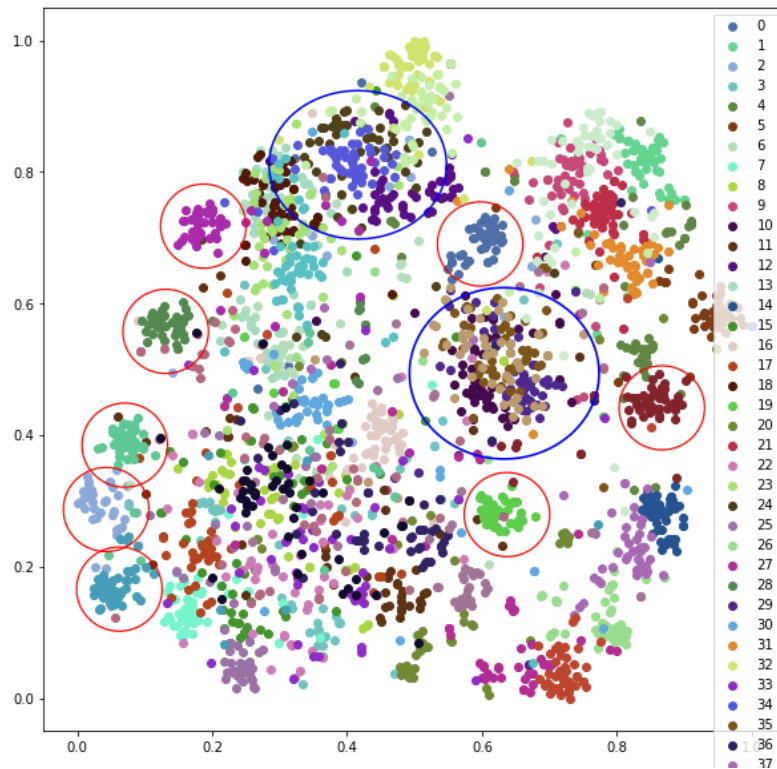
I adopt 101-Layer ResNet[1] and changed the number of the perceptron of output fc layer for our task. Since the model takes 8 pages if directly printed, I adapt the structure from ResNet[1] and make a table as representation. The full printed result could be found in the repository under `p1/Resnet101.txt`.

layer name	output size	structure	
conv1	112×112	$7 \times 7, 64$, stride 2	
conv2.x	56×56	3×3 max pool, stride 2	
conv2.x	28×28		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3.x	14×14		$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4.x	7×7		$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
conv5.x	7×7		$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
output	1×1	average pool, 50-d fc, softmax	

1.2 Accuracy on Validation Set

The accuracy on the public validation set is 0.8748 (2187/2500) ensembling 5 epochs with the least losses during the training phase.

1.3 Visualization with t-SNE



We could observe that some dots with same colors are scattered into distinct groups (those **red circles**). However, those in **blue circles** can not be recognized clearly and hence are predicted wrong at a higher probability. For example, `class 35` and `class 41` are close and ambiguous on the plot, and opening the pictures, we found out that they are pictures of men and women, respectively. As they are alike (having faces, eyes, etc.), visualization shows the similarity between these features.

2 Semantics Segmentation

2.1 Baseline Model Structure

```
FCN32s(  
  (vgg): 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=(1, 1), stride=(1, 1))  
    (1): ReLU(inplace=True)  
    (2): Dropout2d(p=0.5, inplace=False)  
    (3): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))  
    (4): ReLU(inplace=True)  
    (5): Dropout2d(p=0.5, inplace=False)  
  )  
  (score_fr): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))  
  (upscore): ConvTranspose2d(7, 7, kernel_size=(64, 64), stride=(32, 32), bias=False))
```

2.2 Improved Model Structure







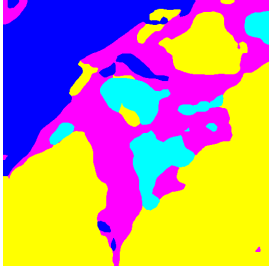
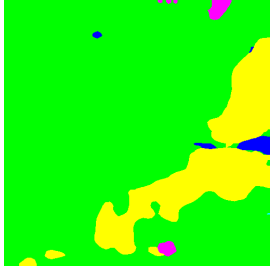

I adopt Deeplabv3[3] introduced by Chen et al. with ResNet101[1] backbone who scores near the SOTA techniques[4] in semantic segmentation task. However, the model printed takes 8 pages so I put the printed result in my repository under `p2/Deeplabv3_Resnet101.txt`

2.3 Mean Intersection-Over-Union

	VGG16_FCN32s	Deeplabv3_ResNet101
mIoU	0.693635	0.734370
epoch	25	25

Notice that the improved model are trained without pretraining.

2.4 Prediction from each Phase

epoch	0010	0097	0107
1			
16			
25			

References

- [1] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
- [2] V. Badrinarayanan, A. Kendall, R. Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. arXiv:1511.00561 [cs.CV]
- [3] Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking Atrous Convolution for Semantic Image Segmentation. arXiv:1706.05587 [cs] (Dec 2017)
- [4] <https://paperswithcode.com/sota/semantic-segmentation-on-cityscapes>