DLCV Homework 2

張緣彩 R07922141

3/11/2019

Collaborators

1. 林子淵 R07921100
2. 黃孟霖 R07922170
3. 楊之郡 R08922050

Problem 1. Baseline Model

Q1. Describe how you pre-process the data.

Because my baseline model and improved model used pretrained *ResNet18* as features extractor (which trained using ImageNet), so I have to normalize the images using mean and standard deviation of ImageNet (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]). Then convert the image vectors into tensors. Finally random horizontal flip the images and segmentation map using hflip from *torchvision.transforms.functional*. My pre-process pipeline of the data are 1) To tensor, 2) Normalization, 3) Random horizontal flip. I only use **random horizontal flip** listed in above for data augmentation.

Q2. Show the following two figures.

1. Training loss versus number of training iterations. (Y coordinate: training loss. X coordinate: number of iterations.)

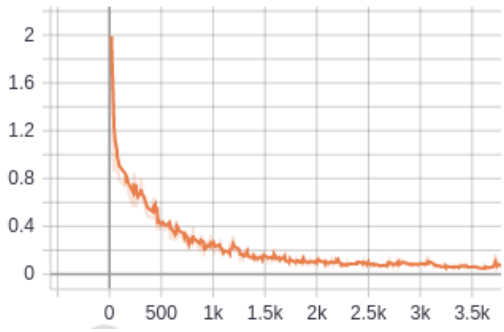


Figure 1: The training loss of baseline model.

2. IoU score on validation set versus number of training iterations. (Y coordinate: IoU score on validation set. X coordinate: number of epochs.)

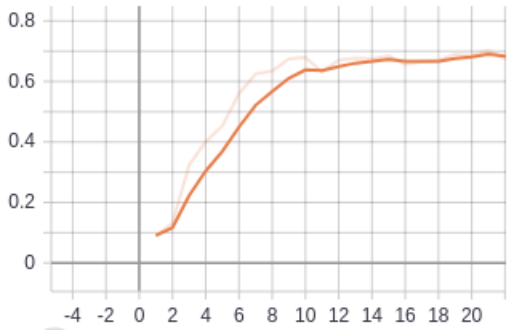


Figure 2: Validation mean IoU score of baseline model.

Q3. Visualize at least one semantic segmentation result for each class.

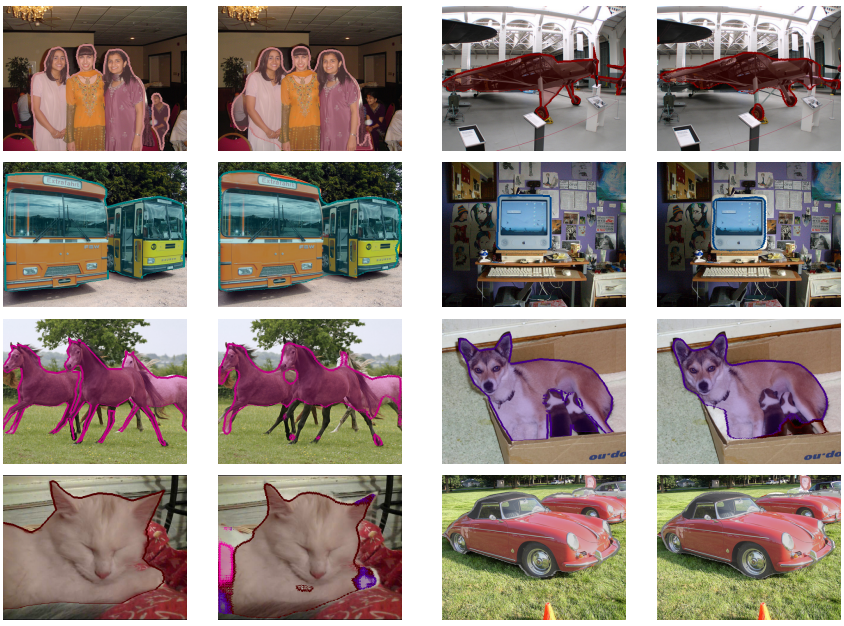


Figure 3: Segmentation maps of baseline model. On the right are ground truth segmentation maps and on the left are baseline model segmentation maps. There are 9 classes, **background**, **person**, **aeroplane**, **bus**, **tv/monitor**, **horse**, **dog**, **cat**, and **car**.

Q4. Report mIoU score and pre-class IoU score of the baseline model. Which class has the highest IoU score? Which class has the lowest IoU score? Please also hypothesize the reason why.

	Validation set
Class 0	0.91298
Class 1	0.75868
Class 2	0.72078
Class 3	0.74666
Class 4	0.40636
Class 5	0.60512
Class 6	0.69490
Class 7	0.77427
Class 8	0.71097
Mean IoU	0.703414

Table 1: Mean IoU and Pre-class IoU score of baseline model on validation set.

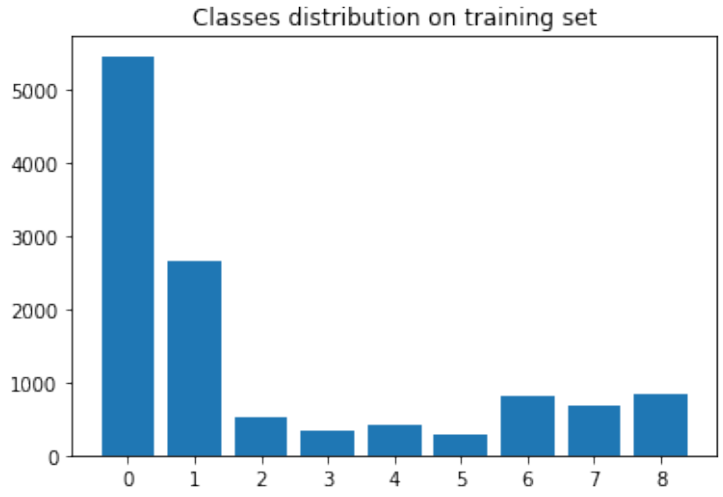


Figure 4: The class distribution in training set. There are 9 classes, **background**, **person**, **aeroplane**, **bus**, **tv/monitor**, **horse**, **dog**, **cat**, and **car**, respectively.

Figure 5: Diversity of Tv/Monitor class in training data.

Class 0 (background) has the highest IoU score and class 4 (tv/monitor) has the lowest IoU score. I think class 0 has the highest IoU score because every training data has background and class 4 has the lower IoU score because 1) number of training data of tv/monitor class is relative fewer than other classes (shown in figure 4), 2) the training image of tv/monitor class has more variety than other class since the screen of the monitor may have other class like cat, dog or human (shown in figure 5). So it is hard for model to capture the features of tv/monitor.

Problem 2. Improved Model

Q1. Draw the model architecture of your improved model.

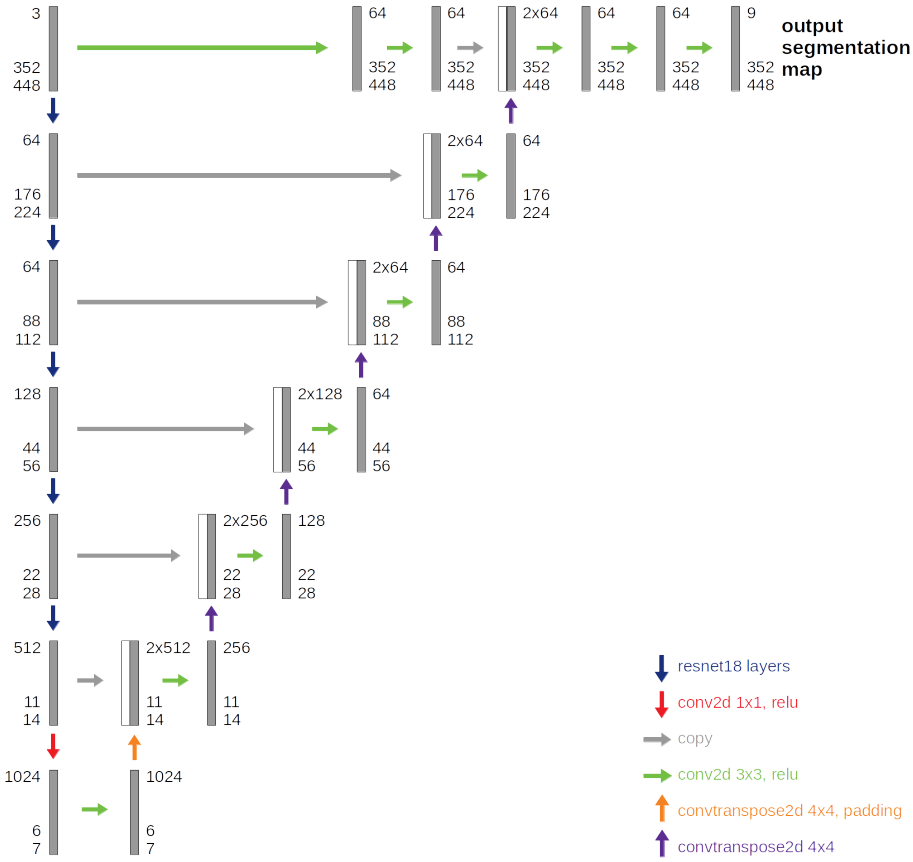


Figure 6: The improved model has the similar architecture as UNet with pretrained ResNet18 as features extractor (encoder).

The design of my improved model is referred from a pytorch-unet GitHub[1]. I used pretrained ResNet18 as features extractor (encoder) same as the GitHub, and added a layer of convolution at last layer of features extractor. Then I changed the whole architecture of the up sampling (decoder) part. I used *ConvTranspose2d* instead of *UpSampling + Conv2d*.
resnet18 layers consists of multiple layers which first layer are *resnet18[0:3]*, second layer are *resnet18[3:5]*, third layer are *resnet18[5]*, fourth layer are *resnet 18[6]*, and fifth layer are *resnet18[7]*. With pretrained weight.
conv2d 1x1, relu block consists of *Conv2d + ReLU + BatchNorm2d*, the parameters of *Conv2d* are kernel size of 1, padding of 0 and stride of 2.

conv2d 3x3, relu block consists of *Conv2d + ReLU + BatchNorm2d*, the parameters of *Conv2d* are kernel size of 3, padding of 1 and stride of 1.
convtranspose2d 4x4, padding block consist of *ConvTranspose2d + ReLU + BatchNorm2d*, the parameters of *ConvTranspose2d* are kernel size of 4, padding of (2, 1), stride of 2, and output padding of (1, 0).
convtranspose2d 4x4 block consist of *ConvTranspose2d + ReLU + BatchNorm2d*, the parameters of *ConvTranspose2d* are kernel size of 4, padding of 1, and stride of 2.

Q2. Discuss the reason why the improved model performs better than the baseline one. You may conduct some experiments and show some evidences to support your discussion.

	Mean IoU
Improved model	**0.766502**
rm0	0.753498
rm1	0.756244
rm2	0.748637
rm3	0.756645
rm4	0.733335
rm5	0.717167
Baseline model	0.703414

Table 2: Mean IoU score on validation set of different model. rmX: remove concatenation of layer 0 to layer X (top layer to lower layers).

I think improved model (UNet) is better than baseline model because when doing up-sampling UNet architecture take same-level feature maps in consideration (concatenate encoder features and decoder features) since both model use the same feature extractor *ResNet18*. To prove this hypothesis, I conducted an experiment. In this experiment, I remove the concatenation in the improved model from layer 0 to layer 5 one by one and only use the remain up-sampling features. I retrain each model with the same hyper-parameters (learning rate: 3e-5, weigh decay: 2e-6, batch size: 32, random seed: 42) for training improved model except it only train for 50 epochs. The validation mIoU showed in the table above. Basically *rm5* is equally same as baseline model but one layer deeper, so that we can consider *rm5* is better than baseline model, and if *rmX* $>$ *rm5* (X $<$ 5), then improved model $>$ *rmX* $>$ baseline model (from table 2). So we can conclude that taking same-level feature maps advantages could really help model in doing segmentation.

Q3. To prove that your improved model is better than the baseline one, report the mIoU score of your improved model. Please also show some semantic segmentation results of your improved model and the baseline model.

	Mean IoU	**Loss**
Baseline Validation	0.703414	0.3829810
Improved Validation	**0.766502**	**0.2870441**

Table 3: Mean IoU score and Loss of Baseline model and Improved model on validation set.

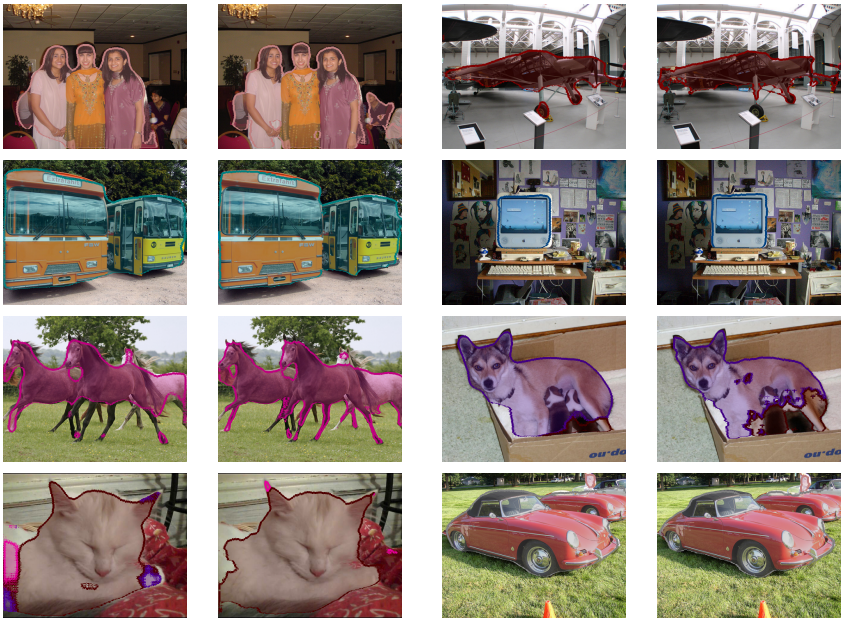


Figure 7: Segmentation maps of improved model. On the right are baseline model segmentation maps and on the left are improved model segmentation maps.

Problem 3. Image Filtering

Q1. Given a variance σ^2, the convolution of a 2D Gaussian kernel can be reduced to two sequential convolutions of a 1D Gaussian kernel. Show that convolving with a 2D Gaussian filter is equivalent to sequentially convolving with a 1D Gaussian filter in both vertical and horizontal directions.

2D Gaussian filter is equivalent to sequentially convolving with a 1D Gaussian filter in both vertical y and horizontal x direction showed in below, $G(x)$ and $G(y)$ are 1D Gaussian filter in horizontal and vertical direction respectively.

$$\begin{aligned} G(x,y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \\ &= G(x) \cdot G(y) \end{aligned} \tag{1}$$

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

$$G(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}$$

Q2. Implement a discrete 2D Gaussian filter using a 3×3 kernel with $\sigma \approx \frac{1}{2ln2}$. Use the provided lena.png as input, and plot the output image in your report. Briefly describe the effect of the filter.

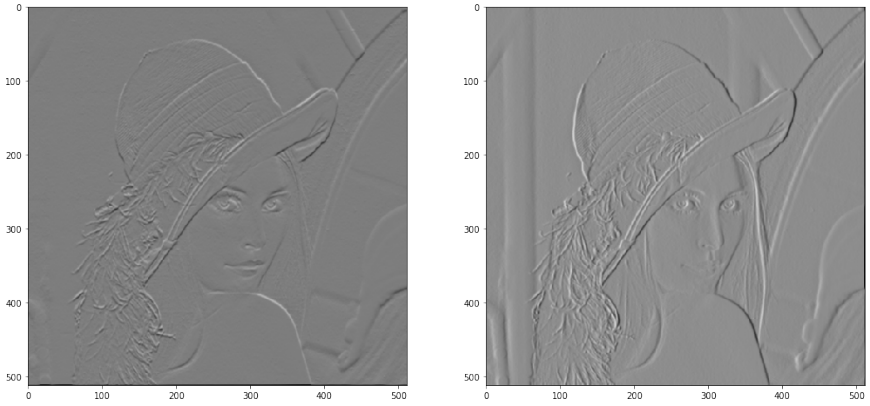
(a) Original lena. (b) Gaussian-filtered lena.

Figure 8: Image of lena.png and Gaussian-filtered lena.

The Gaussian's filter smoothed the image by reducing noise using neighbor value. By comparing these two images, we can see that the original lena has much more noise, and by applying Gaussian's filter on the image, it become much smoother.

Q3. Write down your answers of k_x and k_y. Also, plot the resulting images I_x and I_y using the provided lena.png as input.

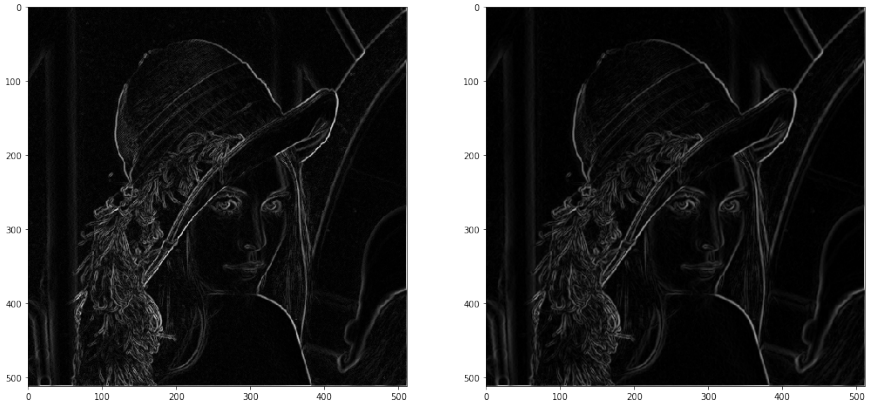
$$\begin{aligned} k_x &= \frac{1}{2}[-1, 0, 1] \\ &= [-0.5, 0, 0.5] \\ k_y &= \frac{1}{2}[[-1], [0], [1]] \\ &= [[-0.5], [0], [0.5]] \end{aligned} \tag{2}$$



(a) $I_x = I * k_x$. (b) $I_y = I * k_y$.

Figure 9: Image of I_x and I_y.

Q4. Use both the provided lena.png and the Gaussian-filtered image you obtained in 2. as input images. Plot the two output gradient magnitude images in your report. Briefly explain the differences in the results.



(a) Without Gaussian's filter. (b) With Gaussian's filter.

Figure 10: Image of lena with and without Gaussian's filter.

The gradient magnitude of Gaussian's filtered lena is much smoother than the original I think it is because Gaussian's filter eliminate noise. I used *scipy.signal.convolve2d* package[2] with 3×3 Gaussian kernel to perform Gaussian filter to the noisy lena image. By comparing the gradient magnitude we can see that gaussian filtered lena has much less noise while the other lena is full of noise gradient magnitude.

References

[1] `https://github.com/usuyama/pytorch-unet/blob/master/pytorch_resnet18_unet.ipynb`

[2] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html`