National Tsing Hua University

Department of Electrical Engineering

EE662000 Computational Photography(計算攝影學), Spring 2019

Homework Assignment #3 Super-resolution

Assigned on Apr 23, 2019

Due by **May 7**, 2019

Python version: 2.7

Assignment Description

Super-resolution is a class of techniques which can enhance the resolution of image. In this assignment, you will implement two different types of super-resolution algorithms. In part1, continued from #HW2, we will use "Proximal library [1]" for super-resolution, including single and multiple image methods. And in part2, we will use Convolutional network (ConvNet) for it. In addition, we will compare and analysis the results from different parameters setting in part1, part2 and with traditional bi-cubic interpolation.

I. Using Proximal to implement Super-resolution

1. SR by using a single image (15%)

A super-resolution problem can be modeled as the following formula (1).

$$\boldsymbol{Y = DBMX + n} \ldots\ldots \textit{(1),}$$

Where,

Y: Original downsized image

D: Downsampling rate

M: Motion shift

B: Gaussian blur kernel

X: Predicted high resolution image

And the problem can be solved iteratively by updating $\boldsymbol{X}$ using the formula (2) in "Proximal."

$$\boldsymbol{Error(x) = DBMX - Y + n} \ldots\ldots \textit{(2),}$$

In each iteration, $\boldsymbol{X}$ is updated with the filtered result of previous iteration, and its initial $\boldsymbol{x_0}$ is equal to upsampled image by using cubic interpolation. Besides, there is only one single image, the Motion($\boldsymbol{M}$) is not taken into consider in this part.

Problem Definition:

$$E(X) = \underbrace{\|D(X \otimes B) - Y\|}_{data\ term} + \underbrace{\lambda \|\nabla X\|_{tv}}_{regulization\ term}$$

$$\|\nabla X\|_{tv} = \sum_p \sqrt{X(p)_x{}^2 + X(p)_y{}^2}$$

However, the center point of the filter kernel varies in different resoltion images, we will need to shift the center point of Gaussian blur kernel ***K*** for the sake of the alignment of downsized image and the Gaussian blur kernel. The detailed flow of the center point shifting is shown in Figure 2. By shifting the center before filtering, we can align the center of Gaussian blur kernel with the center point of down sized image when doing convolution.

***K*: Gaussian blur kernel is defined as formula (3), and its parameter setting is the same as Table. 1.**

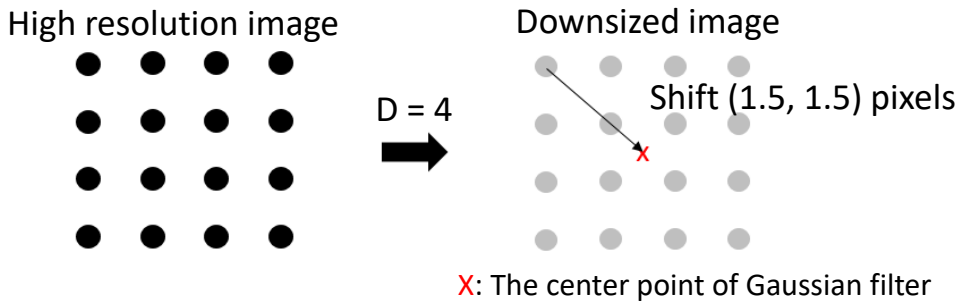


Figure 1. The center point of image will be shifted after downsampling.

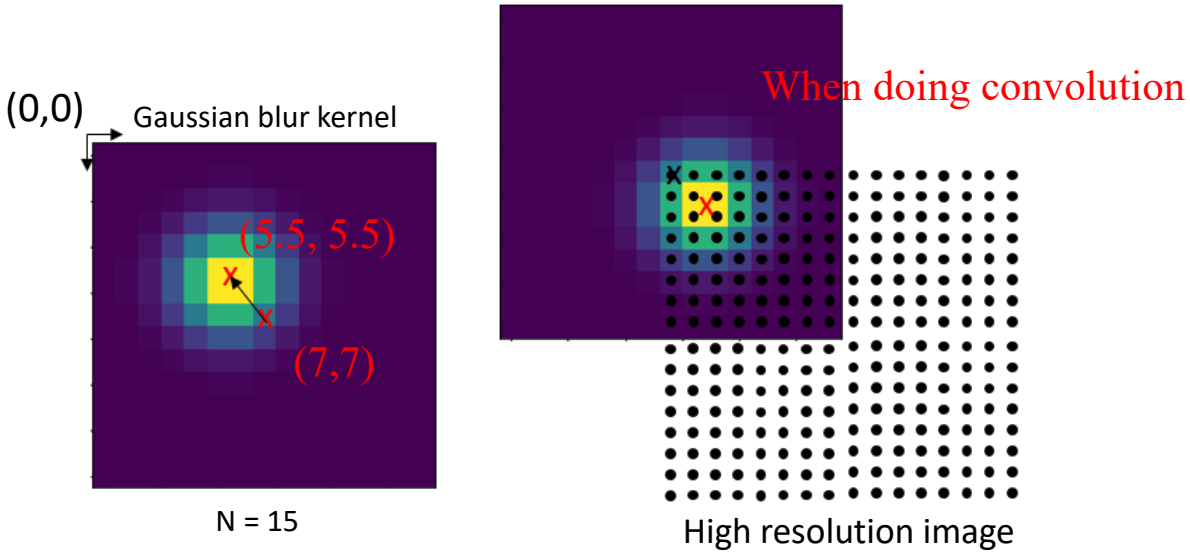


Figure 2. The reason why you should shift the center of Gaussian blur kernel.

$$\text{Gaussian filter(i, j, mid\_x, mid\_y)} = \frac{\exp\left(-\frac{(i-mid\_x)^2+(j-mid\_y)^2}{2\sigma_s{}^2}\right)}{\sum_{(i,j)} Gaussian\_filter(i,j)} \quad \ldots\ldots (3)$$

mid_x = ($\left\lfloor \frac{N}{2} \right\rfloor$ – 1.5), mid_y = ($\left\lfloor \frac{N}{2} \right\rfloor$ – 1.5), when D = 4 and N = 15.

If D = 4, the shift caused by downsample is equal to 1.5.

Table 1. Parameter setting in question 1.

	Gaussian blur kernel		Problem definition
Parameter	N	σ_s	λ
Value	15	1.5	0.01

Your result should be similar to "zebra_test_sr_single.png" in /Ref (PSNR > 30dB) and you will get a full score, when your parameter setting is same as table 1.

2. SR by using multiple image (25%)

In this part we will focus on reconstructing the high resolution image with multiple low resolution images of same objects. This type of method is especially useful when the observed source is a sequence images of target scene. The problem definition of multiple images method is expressed in the following:

Problem definition:

$$\boldsymbol{E_{total} = \underbrace{E_1 + E_2 + E_3 + E_4}_{data\ term} + \underbrace{\lambda \|\nabla X\|_{tv}}_{regulization\ term}}$$

$$\|\nabla X\|_{tv} = \sum_p \sqrt{X(p)_x{}^2 + X(p)_y{}^2}$$

$$\mathbf{E}_n = \|D(X \otimes B_n) - Y_n\|,\ n = 1, 2, 3, 4$$

Where,

$\mathbf{Y}_n$***: Original downsized image***

D: Downsampling rate

$\mathbf{B}_n$***: Gaussian blur kernel***

X: Predicted high resolution image

Assume source input is a set of four images, and each one has a motion shift to the others. As a result, the center point of the Gaussian kernel should be different from each other. For example, if an image has motion shift: (mv_x, mv_y) = (-0.5, 0) according to the original low resolution image, it will be enlarged to be (-2, 0) after upsampling. And as mentioned in part I question 1, the center point of Gaussian filter will vary in different resolution. Combining the effects of motion shift and resolution, you have to change the center point of Gaussian filter. The

final displacement can be expressed as formula (4). As shown in Figure 3, using D = 4, and motion shift (-0.5, 0) as example, the final shift (x, y) is ($\underbrace{1.5}_{Resolution}$ + $\underbrace{2}_{Motion\ shift}$, $\underbrace{1.5}_{Resolution}$) pixels.

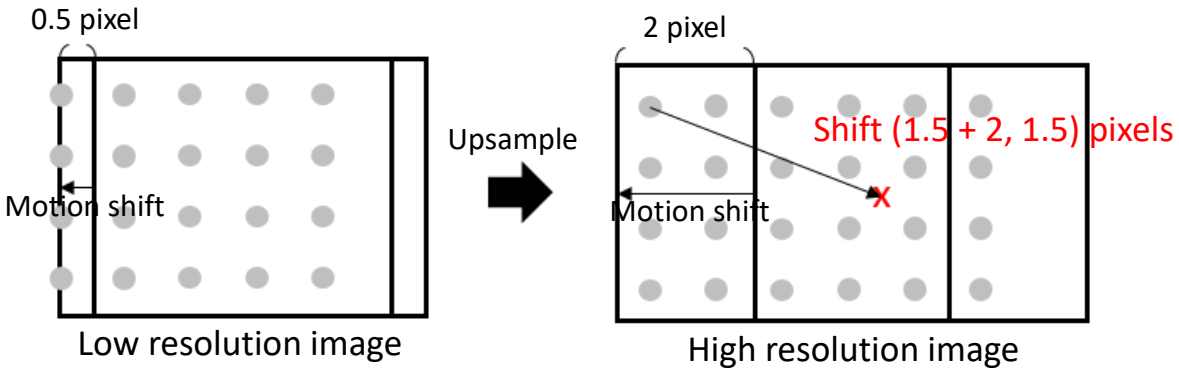


Figure 3. The center point will be changed when adding the motion shift.

Adjusted Gaussian filter' s center point:

$$\textbf{Mid\_x} = \left\lfloor \frac{N}{2} \right\rfloor \textbf{ – 1.5 – motion shift\_x * D,}$$

$$\textbf{Mid\_y} = \left\lfloor \frac{N}{2} \right\rfloor \textbf{ – 1.5 – motion shift\_y * D …… (4)}$$

If D = 4, the shift caused by resolution is equal to 1.5.

Table 2. The motion shift of each image in low resolution images.

	Image No .1	Image No .2	Image No .3	Image No .4
mv_x(pixel)	0	-0.5	0	-0.5
mv_y(pixel)	0	0	-0.5	-0.5

Table 3. Parameter setting in question 2.

	Gaussian blur kernel		Problem definition
Parameter	N	σ_s	λ
Value	17	1.5	0.01

Your result should be similar to "zebra_test_sr_four.png" in /Ref (PSNR > 30dB), and you will get a full score, when your parameter setting is same as table.3.

II. Using ConvNet to implement Super-resolution

In this part, we will build a ConvNet for super-resolution using "Pytorch [2]", a python-based package to provide a platform for ConvNet research.

1. Construct the ZebraSRNet (20%)

A popular Network architecture "Residual block"[3] and the corresponding model definition is shown in Figure 4 and 5 respectively. Short-cut connection (or skip connection) in the residual block is the connection which skip one or more layers, and it can avoid the problem of vanishing gradients such that deeper networks are able to be converging state. In this part, we will construct ConvNet architecture called "ZebraSRNet" based on "Residual block". You are asked to construct the "ZebraSRNet" using Pytorch template in "model.py". The detailed architecture of ZebraSRNet is shown in Figure 6.

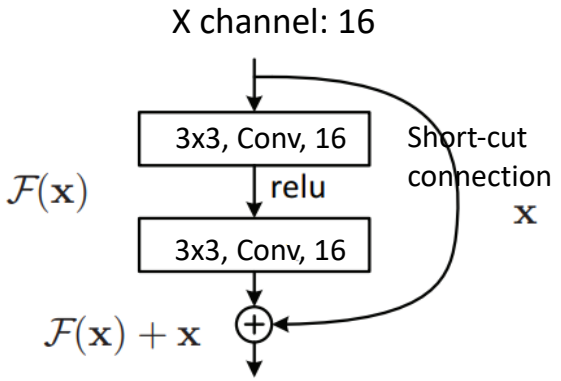
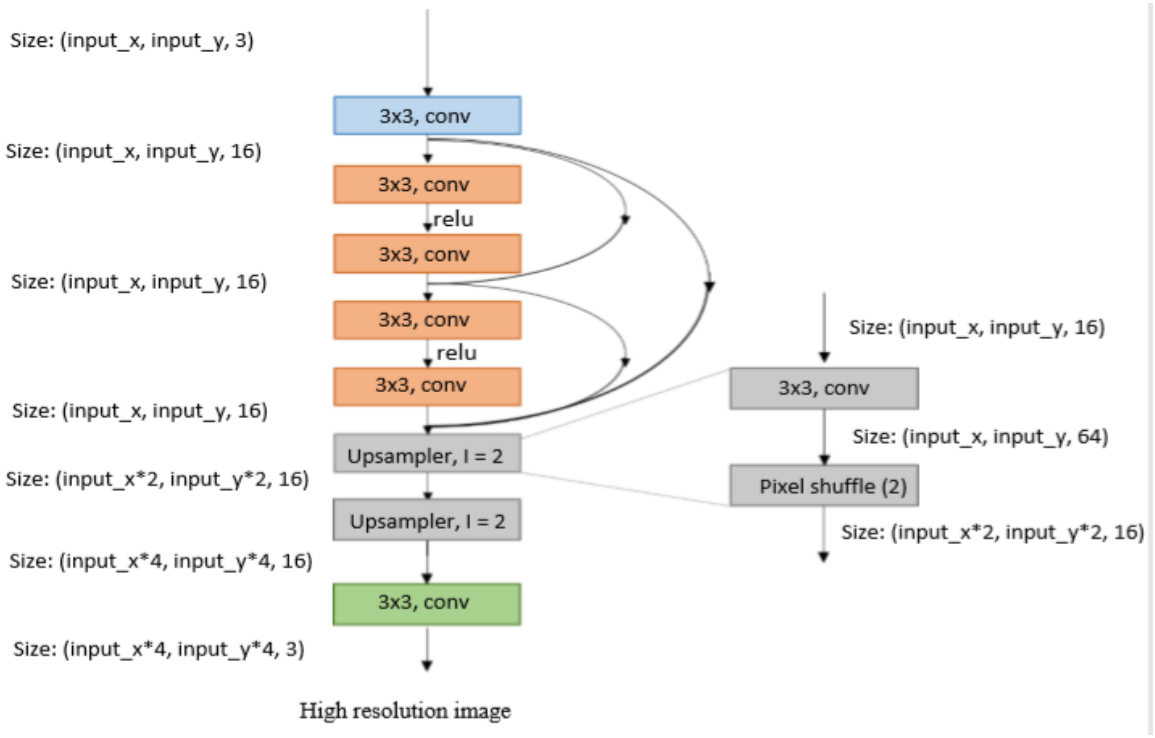


Figure 4. The simple network architecture for example.

```python
class ResBlock(nn.Module):
    def __init__(self, bias=True, act=nn.ReLU(True)):
        super(ResBlock, self).__init__()
        modules = []
        modules.append(nn.Conv2d(16, 16, 3, padding=1))
        modules.append(act)
        modules.append(nn.Conv2d(16, 16, 3, padding=1))
        self.body = nn.Sequential(*modules)
    def forward(self, x):
        res = self.body(x)
        res += x
        return res
```

Figure 5. Example python code for Residual block.



▲Figure .6 The network architecture of ZebraSRNet.

After completing the model definition of ZebraSRNet, you can use the following command to do the training and testing.

For training:

$>> python train.py --nFeat 16 --nResBlock 2 --nEpochs 15

P.S. If your OS is windows, please add the argument "--thread 0".

And for the testing:

$ python test.py –model model_pretrained/net_F16B2_epoch_15.pth --input_image image_test/LR_zebra_test.png --output_filename result/F16B2_zebra_test.png --compare_image ref/HR_zebra_test.png

Table 4. Parameter setting in part II question 1.

Parameter	nFeat	nResblock	nEpochs
Value	16	2	15

Your result should be similar to "HR_zebra_test.png.png" in /Ref (PSNR > 15dB), and you will get a full score, when your parameter setting is same as table.4.

2. Increase the size of feature map and the number of Residual block (15%)

Now based on the network architecture of ZebraSRNet in Figure 6, try to increase the size of feature map to **64,** and the number of residual block to **8.** Don't forget to add the short-cut connection from the output of head Conv layer to the input of first upsampler layer.

▲Table. 5 Parameter setting in part II question 2.

Parameter	nFeat	nResblock	nEpochs
Value	64	8	100

Your result should be similar to "HR_zebra_test.png.png" in /Ref (PSNR > 20dB), and you will get a full score, when your parameter setting is same as table 5.

III. Report (25%)

1. **(5%)** Test different values of lambda and gaussian std to see the difference of results in **Part I**, and compare and explain your result with "image/HR_zebra_test.png".
2. **(5%)** Compare subjective and objective quality for different interpolation results: bi-cubic, TVL1 (one image), TVL1 (four images), ZebraSRNet-F64B8.
3. **(5%)** Test the hidden image in the "/image_hidden", and explain what happens to results and try to improve them by using ZebraSRNet-F64B8.Run the following command:

   $>> *python test.py --model model_pretrained/net_F64B8_epoch_100.pth --input_image image_hidden/LR_panda.png --output_filename result/F64B8_panda_test.png*
4. **(10%)** Use your own picture to do Super-resolution by using ZebraSRNet-F64B8. Discuss how do you do in the data preprocess and show the result of testing. **Remember that training image should be different to testing image.**

IV. Deliverable and file organization

Directory	Filename	Description
hw3/part1	result/*.png	Your result
	*.py	All the python code
hw3/part2	image_hidden/*.png	Specific images for testing
	ref/*.png	Reference answer
	model_pretrained/net_F16B2_epoch_15.pth	Training result for F16B2
	model_pretrained/net_F64B8_epoch_100.pth	Training result for F64B8
	result/*.png	Your result
	*.py	All the python code
hw3/report	report.pdf	Your report\
	own_image/*.png	Your own images, include own training/testing data

When you submit the #HW3, please organize your files carefully according to the above table and compress your files to HW3_studentID in ZIP format. If there is invalid file/data organization, you will get 5% score punishment.

V. Referecne

[1] ProxImaL http://www.proximal-lang.org/en/latest/

[2] Pytorch http://pytorch.org/tutorials/

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.