National Tsing Hua University
Department of Electrical Engineering
EE6620 Computational Photography, Spring 2022

Homework #3
Super Resolution

Assigned on April 22, 2022

Due by May 13, 2022

Homework Description

Super-resolution is a class of techniques which can enhance the resolution of images. In this assignment, you will work on two types of super-resolution methods, **optimization-based** and **convnet-based**. For the optimization-based part, you will solve the SR problem with a single-image method and multi-image method. For the convnet-based part, you are going to build and train a SR model, then try to increase its size.

Low-resolution image

Optimization-based SR

Convnet-based SR

Table 1: Scores

Items		Score
Implementation (50%)	Single-Image SR	10%
	Multi-Image SR	15%
	Construct SR Model	20%
	Increase Model Size	5 %
Report (50%)	Experiments	30%
	Free Study	20%

1 Implementation

In this section, you will work on the following four problems. Your results should be saved in /result, and we will evaluate your results based on the PSNR metric. The high resolution images and TA's reference answers are provided in /reference for evaluation.

1.1 Single-Image Super Resolution (10%)

Super-resolution is an inverse problem and can be modeled as

$$Y = DBMX + n, \tag{1}$$

where Y is the observed low-resolution image, X is the latent high-resolution image, M is motion shift, B is blur, D is down-sampling and n is noise. We will use ProxImaL package [1] as in homework 2 to solve the latent image X. The objective function for single-image SR problem is defined as

$$E(X) = ||D(X \circledast B) - Y|| + \lambda ||\nabla X||_{tv}, \tag{2}$$

$$||\nabla X||_{tv} = \sum_p \sqrt{X(p)_x^2 + X(p)_y^2}. \tag{3}$$

Specifically, Y is a provided low-resolution image, D is down-sampling by four, B is a Gaussian kernel and M is omitted since there is only one single image. The Gaussian kernel here is

$$B_{Gaussian}(C_x, C_y, k, l) = \frac{\exp\left(-\frac{||(C_x, C_y)-(k,l)||^2}{2\sigma_s^2}\right)}{\sum_{k,l} B_{Gaussian}}, \; k, l \in [-N/2, N/2] \tag{4}$$

where (C_x, C_y) is the kernel center index, and is equal to (-1.5, -1.5) here, which means the kernel has a center shift away from (0, 0).

The down-sampling by four process is that we sample one pixel form every four pixels along each dimension. Here we assume that each pixel in LR images is sampled from the center point of every 16 pixels in HR images. However, in actual implementation, *e.g.*, `subsample()` function provided in ProxImaL, it sample the first pixel (the upper left pixel) from every 16 pixels. The two different down-sample schemes are shown in Figure 1.Thus when convolving with the first pixel, the center point of kernel should align to the center point of the 16 pixels to match our assumption, which is illustrated in Figure 2. In the case of down-sampling by four, the kernel center shift would be (+1.5, +1.5) pixels. The reason why the actual center shift is (-1.5, -1.5) pixels in Eq. (4) is that when doing convolution, we should flip the kernel beforehand.

In this part, you need to complete the Gaussian kernel function `get_kernel()` in sr_single.py. Your result for /image/LR_zebra_test.png should be similar to /reference/zebra_test_single_golden.png (PSNR>60dB), under the default setting`{N,` σ_s`,` λ`}={11, 1.2, 0.01}` to get a full score.

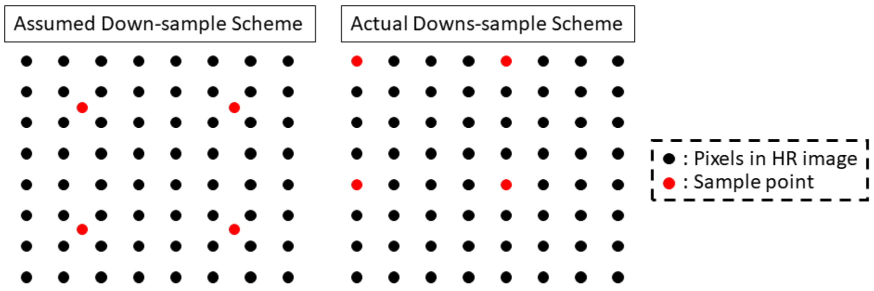

Figure 1: Different down-sample schemes

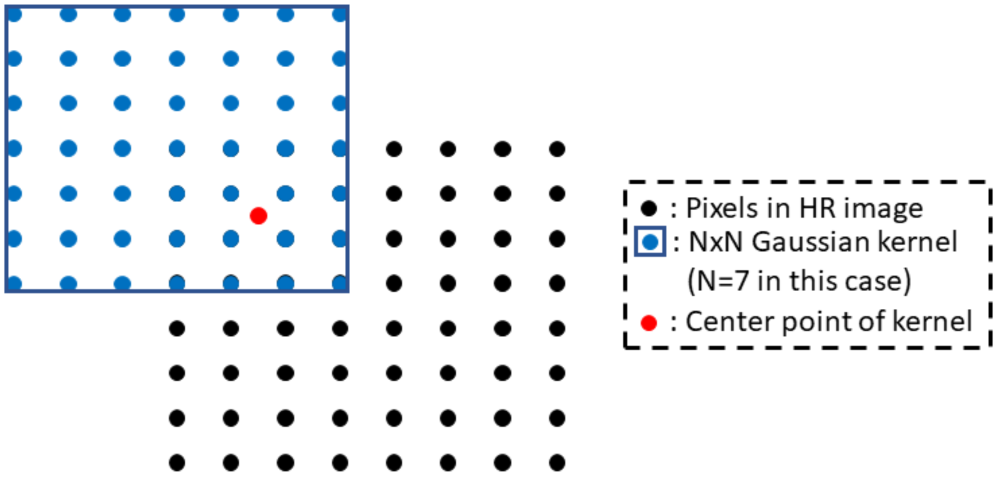
● : Pixels in HR image
: NxN Gaussian kernel (N=7 in this case)
● : Center point of kernel

Figure 2: Alignment of kernel center when doing convolution

1.2 Multi-Image Super Resolution (15%)

Section 1.1 use one single LR image as the data. In this part, we provide **eight** LR images Y_i and they have motion shift relative to each other. The objective function for multi-image SR is defined as

$$E(X) = (\sum_i ||D(X \circledast B_i) - Y_i||) + \lambda||\nabla X||_{tv},\ i \in [0,7] \tag{5}$$

$$||\nabla X||_{tv} = \sum_p \sqrt{X(p)_x^2 + X(p)_y^2}. \tag{6}$$

The effect of motion shift would be combined in the Gaussian kernel B_i, that is, you should further adjust the center point of each kernel based on the relative motion shifts. The kernel center due to both down-sampling and motion can be summarized as

$$C_x = -1.5 + 4 * mv_x, \tag{7}$$

$$C_y = -1.5 + 4 * mv_y, \tag{8}$$

where mv_x and mv_y are the relative motion shifts in LR images.

In this part, you need to complete the Gaussian kernel function `get_kernel()`, and formulate the optimization problem in `solve()` in sr_eight.py. Your result for /image/LR_zebra_test.png should be similar to /reference/zebra_test_eight_golden.png (PSNR>60dB), under the default setting{`N`, σ_s, λ}={`13, 1.2, 0.01`} to get a full score.

1.3 Construct SR Model (20%)

In this part, we will use PyTorch library [2] as a machine learning framework that helps us building and training convnet models. A popular SR model called WDSR is built upon WDSR block, which has a short-cut connection across one or more layers. Example WDSR blocks including WDSR block type A and WDSR block typeB are shown in Figure 3. C stands for base channel number(nFeat), and R stands for expansion ratio(ExpandRatio). Note that we will set the closest integer as channel number if 0.8*C is not an integer. To learn the basic usage of PyTorch library, you can refer to the example code `WDSRblock_typeA` in model.py for WDSR block typeA.

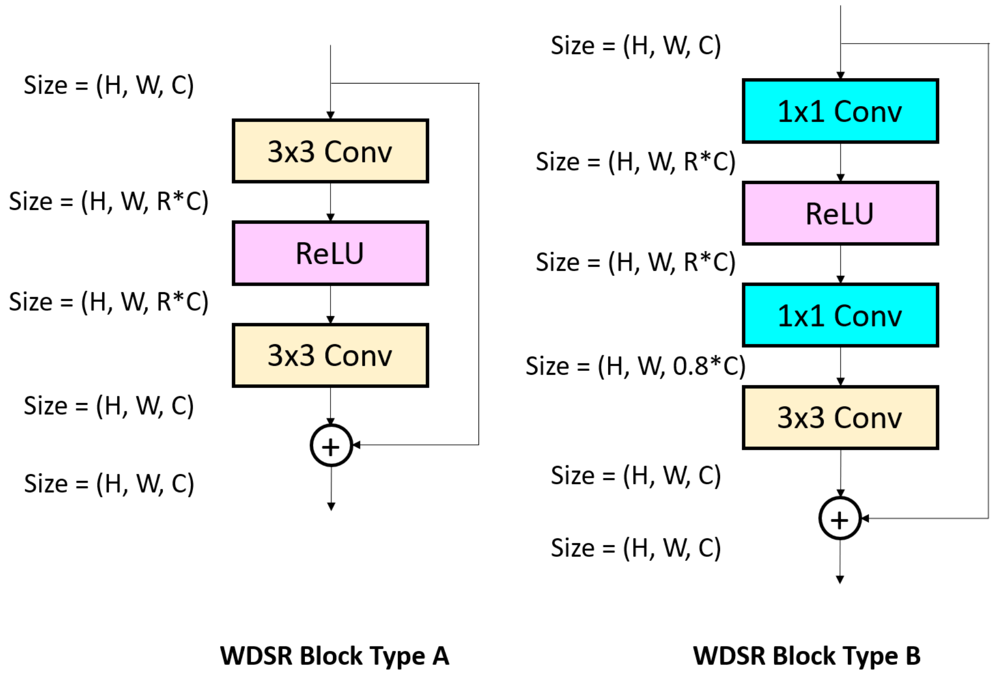

Figure 3: Example WDSR Blocks.

The network architecture of our model called ZebraSRNet is shown in Figure 4. You need to construct this model by completing the class definition of `WDSRblock_typeB`, `upsampler` and `ZebraSRNet` in model.py. Note that we choose the WDSR block type B for its smaller parameter size. After finishing constructing the model, you can train this model by typing the following command in terminal.

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

If you have GPU, you can add the argument **<u>–cuda</u>** for faster training, and if you use Windows as your operating system, you need to add the argument **<u>–thread 0</u>**. For testing, run the following command.

```
$ python test.py --model_path model_trained/net_F16B2E2_epoch_15.pth --input_image
    _path image_test/LR_zebra_test.png --output_image_path result/F16B2E2_zebra_
    test.png --compare_image_path reference/F16B2E2_zebra_test_golden.png
```

Your result for /image_test/LR_zebra_test.png should be similar to /reference/F16B2E2_zebra_test_golden.png (PSNR>30dB), under the setting{nFeat, ExpandRatio, nResBlock, nEpochs}={16, 2, 2, 15} to get a full score.

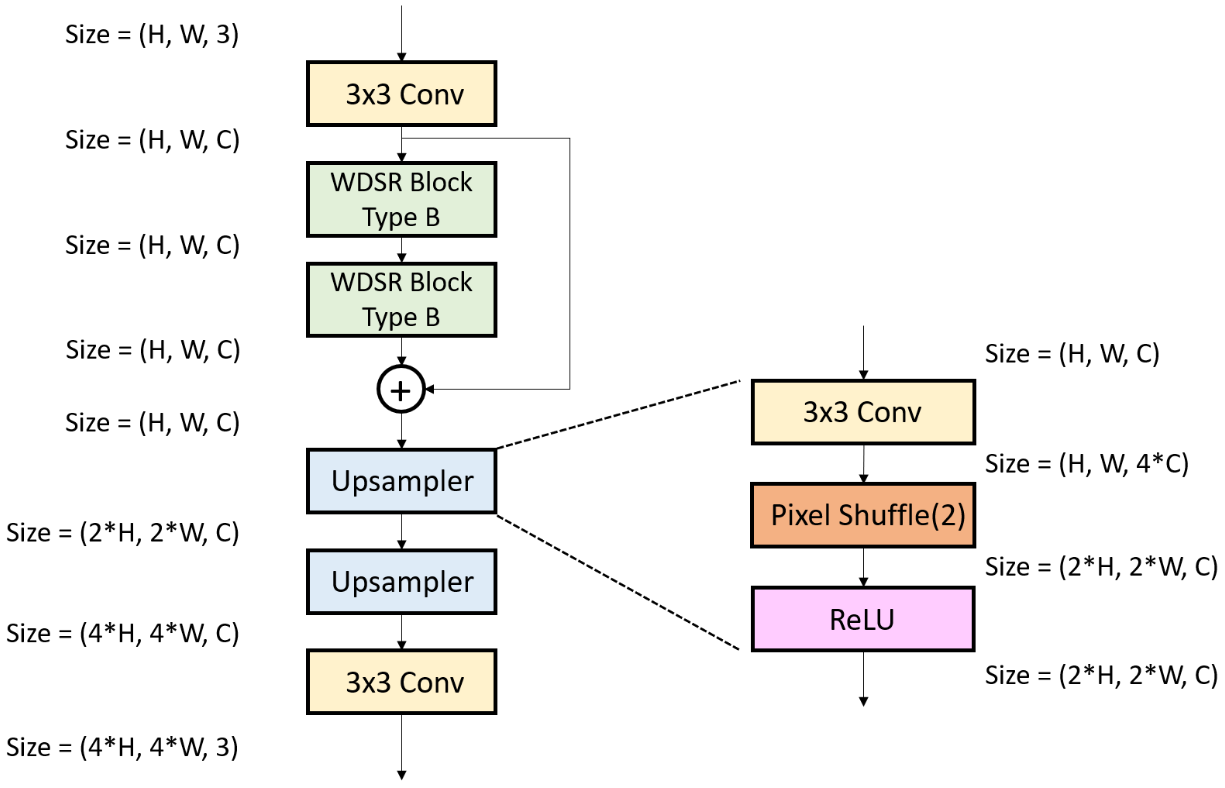

Figure 4: Network architecture of ZebraSRNet

1.4 Increase Model Size (5%)

Based on the model you constructed in 1.3, we can increase the model size by increasing its width and depth. In this part, you need to modify the arguments of the training command. Your result for /image_test/LR_zebra_test.png should be similar to /reference/F32B8E4_zebra_test_golden.png (PSNR>30dB), under the setting{nFeat, ExpandRatio, nResBlock, nEpochs}={32, 4, 8, 120} to get a full score.

2 Experiments & Free Study (50%)

There are two parts of problems: Experiments and Free Study. Finish problems in experiments and choose at least two interesting topics of super-resolution for your free study. You should answer the problems by following the two-step research flow: **Assumption** and **Justification**. Based on your observation, repeatedly propose an assumption, and justify your idea by experiments. After you reach a satisfying result, clearly describe your ideas and justification in the report. Grading criteria for this part are based on the clarity of the report and rigorousness of the justification. Note that you can discuss as more topic as you want in free study; however, we will consider both breadth and depth in grading, and **2% and 1% bonus scores on semester will be given to the best and the second-best stduents**.

2.1 Experiments (30%)

a. (10%) In 1.2, we use eight images to solve the SR problem. What if we use fewer images? Is that using more images results in better quality? If not, try to propose assumptions and justify through experiments. You should select two, four, and six images from these eight images and conduct the experiments.

b. (10%) Test the model you trained in both 1.3 and 1.4 with the provided images in image_hidden/. What happen to the results? Why the results look weird? Try to resolve the artifacts and provide justification for how or how not your method work.

c. (10%) Prepare your own training set and testing set by taking your own photos, and then train a model for super-resolution. How do you choose the training set and training flow? How your designed flow achieve your super-resolution task? **Remember that testing set should be different from training set**.

2.2 Free study (20%)

Find at least two interesting aspects of super-resolution as your research topics. You can also choose your research topics from the following problems.

1. Try different parameter settings $\{N, \sigma_s, \lambda\}$ in 1.2. What's the effect of each parameter?
2. Why the kernel center shifts for multi-image SR can be expressed as Eq. (7) and Eq. (8)?
3. Taking your own photos as the data and perform multi-image SR as in 1.2. How do you take those photos? How do you estimate the motion shift of each photo?
4. Compare the outcomes of different SR method, *i.e.*, direct interpolation, optimization-based and convnet-based. What's the pros and cons of different methods? What's the applicable scene of these methods? Try to justify your conclusion with further examples instead of only using test images in this assignment.

5. What's the difference between two quality metrics, SSIM and PSNR? Are these metrics always consistent with human's intuitions?
6. We increase model width and depth in 1.4. Is there any other way to improve the quality of convnet models?

3 Deliverable

- HW3_studentID/
  - optimization-based/
    - sr_single.py
    - sr_eight.py
    - result/
      - zebra_test_single.png
      - zebra_test_eight.png
  - convnet-based/
    - model.py
    - train_net_F16B2E2.log
    - train_net_F32B8E4.log
    - result/
      - F16B2E2_zebra_test.png
      - F32B8E4_zebra_test.png
    - model_trained/
      - net_F16B2E2_epoch_15.pth
      - net_F32B8E4_epoch_120.pth
  - own_image/
    - trainset/*.png (all your training images used in 2.1)
    - testset/*.png (all your testing images used in 2.1)
  - HW3_report_studentID.pdf

Compress your whole folder HW3_studentID/ in HW3_studentID.zip and submit to eeclass.

Note that wrong file delivery or arrangement will get 5% punishment.

References

[1] F. Heide, S. Diamond, M. Nießner, J. Ragan-Kelley, W. Heidrich, and G. Wetzstein, "Proximal: Efficient image optimization using proximal algorithms," *ACM Trans. Graph.*, vol. 35, July 2016.

[2] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison,

A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.