VRDL Homework 4: Image Super Resolution

Yi-Cheng Hung
Department of Applied Mathematics, NYCU

September 18, 2022

1 Introduction

The introduction of this homework is an Image Super-Resolution. This task starts with the resolution problem, returning low resolution to high resolution through reconstruction, and this problem can be divided into two types, namely classical multi-image SR and single-image multi-patch SR. The task is to do the latter. By using the bi-cubic interpolation method to reduce the original photo by three times to simulate a low-resolution photo, also known as a downscaled image, there are six deep learning methods mentioned in class. We use the improved new model for training based on RCAN[1] to reach the baseline. We did the data prepossessing, implement the model. In pros-processing, ensemble. **Github:** https://github.com/YCHung1998/Image-Super-Resolution.git

1.1 Data Exploration

In this task, we have 291 high-resolution (HR) images for training and 14 low-resolution (LR) images in the test data that are downscaled images created by bicubic interpolation.

Down scaled 3x (LR)

High resolution image

1.2 Data Preprocessing

1. Since the data only provides high-resolution images, our goal is to enlarge the low-resolution photos by 3 times, so we first generate the low-resolution images to 1/3 of the original width and height through bicubic interpolation.

2. Each image is a different size. There is a problem of not being divisible by three times. Here, the high-resolution photo is selected with the top left as the starting pixel, and the part that is not divisible by the last column or row is discarded. When calculating the loss, there will be no problems due to inconsistent size, or the PSNR value will drop due to over-calculating the area where the photo is not restored.

3. The data size of the crop is determined by the patch size. Since the low-resolution image may be smaller than the patch size, there are two ways to deal with it. The patch size is smaller, or it is processed through padding. Since we want to fix the patch size not to be too small because of some photos, we choose padding, which needs to be padded to a specified size.

4. Enhanced random rotation and mirror transformation to increase the variety of data.

1.3 Model Selected

The reference paper we chose is Densely Residual Laplacian Super-Resolution. We choose Dense Residual Laplacian Network (DRLN)[2] for two main reasons. First, Densely Residual Network (DRN) aims to learn local features from the images via dense-connections which proposed by Zhang et al. In addition, it also avoids gradient vanishing. The second is because the text mentioned "The PSNR values of RCAN is the best among all the algorithms as mentioned earlier." and this model is based on Residual Channel Attention Network (RCAN)[1] . This method improves upon RCAN both visually and in numbers by exploiting densely connected residual blocks followed by multi-scale attention using different levels of the skip and the cascading connections. And we had learned RCAN in class.

Before the introduction of the model architecture, first review the main architecture of RCAN, which is divided into four parts. And also look the structure below the figure.

1. Shallow feature extraction

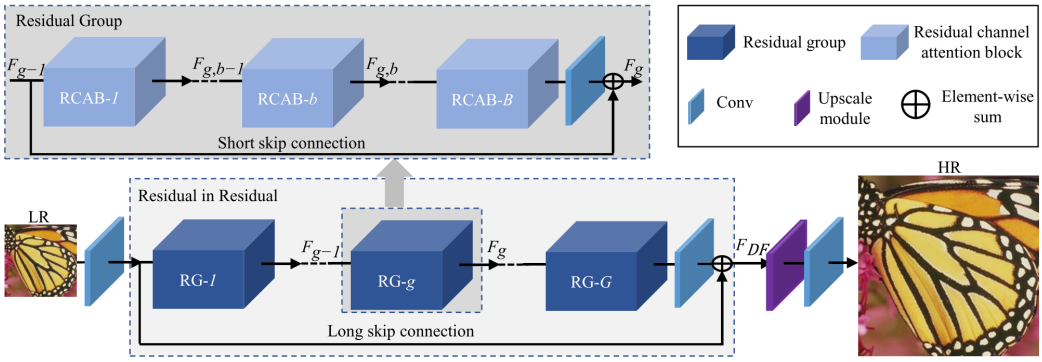
   One convolution layer extract feature from patch. It shows between LR and RIR

below the figure.

2. RIR (residual in residual) deep feature extraction
   Consisted to Residual Group and RG-x is residual learning parts, the main image is translated by the long skip-connections. It provides a large receptive field size.

3. Upscale module
   The corresponding purple layer function is to zoom in to the image size we want.

4. Reconstruction part
   Only one convolution layer with one channel to gray image or multi channel to rgb image.



Network architecture of our residual channel attention network (RCAN)

Figure 1. RCAN model architecture

There are two differences between DRLN and RCAN. Use the dense network connection method. In RIR, RG-x is replaced by DRLM block. Although the name has changed, the action is still a little changed. DRLN uses DRLM instead of RCAN and performs connection layers in each layer.
As shown in the figure below, there are more blue concatenate connections between each DRLM, and Laplacian Attention is responsible for the second half of the DRLM.

- **Dense Residual Laplacian Module (DRLM)**
  Each DRLM consists of a densely connected residual unit, compression unit, and Laplacian pyramid attention unit.

- **Laplacian Attention**
  In super-resolution, the same concept with a little variation can be applied that features should be weighted according to their relative importance. Here use use a global descriptor to capture the statistics expressing the entire image $\mathcal{G}d$. All the activation is ReLU, the convolution layers fixed to 64. In Laplacian attention

where the filter size is 3, 5, 7 in pyramid attention, the feature maps are reduced by factor of 4. In the last activation apply the sigmoid function.

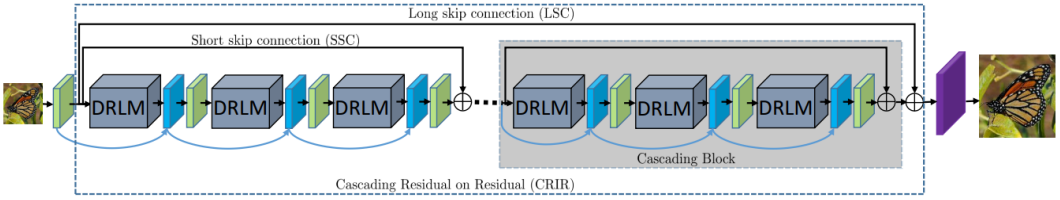
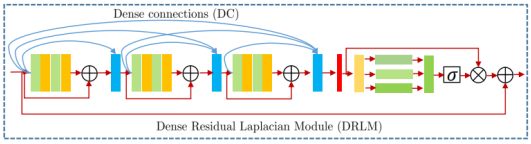
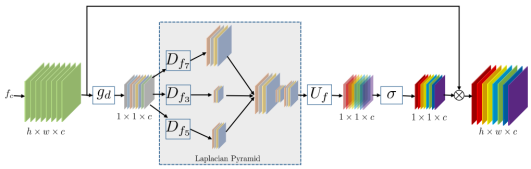


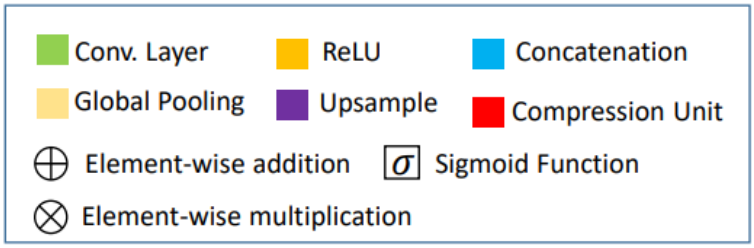
Figure 2. DRLN model architecture



DRLM block

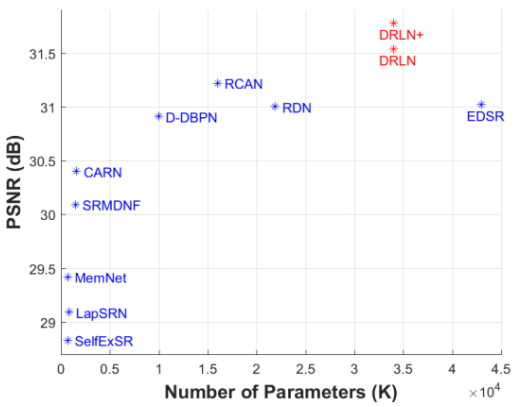


Laplacian attention

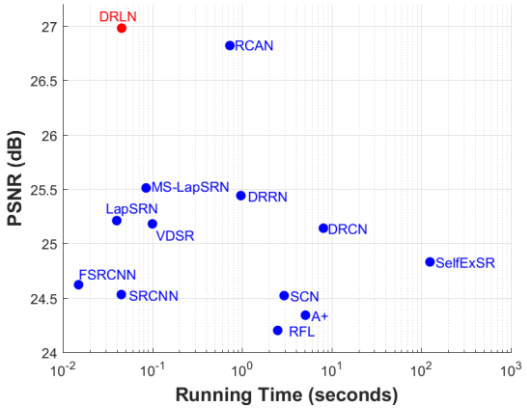


In each cascading residual on the residual block, we have three DRLMs, and in each DRLM, we have three RBs densely connected, one compression unit and one Laplacian attention.

Comparing the amount of parameters and time required for PSNR-based performance according to the figure below, DRLN may be more complex but faster to compute than RCAN.



Parameter vs. performance



Time vs. performance

Loss

Here we use one-norm loss to evaluate the model. W is weight in model DRLN.

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} ||DRLN_W(x_i) - y_i||_1.$$

Initial Setting

The max epoch is 1500 with batch size 16 and patch size 32, loss function is L_1-loss and optimizer is Adam with the setting values below.

- betas = (0.9, 0.999)
- init_lr = 1e-4
- step_size = 100
- gamma = 0.5

2 Experiments

Before the successful training, because the photos placed in the validation were too large, out of memory appeared in the validation stage. Later, the photos of the validation were determined according to the image size of the train data, and it ran smoothly.

Here, we compare the results of individual models under different hyper-parameters. Because the epoch is set to 400, it is found that the loss can be further reduced, so it is adjusted to the upper limit of 1500(ep). In 1500, the learning rate (lr) is reduced to 0.5 times every 50 steps, which will make the later lr too small, so as the total number of epochs increases, it is changed to 100 steps and reduced to 0.5 times. Second, in order for the model to run smoothly on the GPU, there is a trade-off between batch size and patch size. Patches are mostly related to the image itself and depend on the size of each cut. And we found that when running 1500 epochs, the GPU can just run a combination of 20 and 48, but in terms of loss, it hovers around some values around 1000. And taking the best PSNR index also shows that the effects of 16 and 32 are relatively good.

Epoch (ep)	400 (379)	400 (394)	1500 (1185)	1500 (895)	1500 (1054)
Batch size	16	16	16	16	20
Patch size	48	32	48	32	48
Decay step	50	50	100	100	100
PSNR	27.6887	27.7158	27.8898	**27.9725**	27.9320

Table 1. Single model comparison

The previous prediction results just input the original LR photo into the model prediction. Now we put various mirrored and rotated pictures into the model ensemble. The following uses an image example.


original


horizontal (h)


vertical (v)


(h + v)


rotation 90°


rotation 180°


(r90°+v)


(r90°+h)

Divide the original image from the table below, add mirror transformation, Adding rotation transformation and mirror rotation are added to compare the PSNR values, and it will be better to show the model in different directions.

(Batch, patch)	(16, 32)		(20, 48)	
Transform \ Epoch	895	1239	895	1054
original	27.9725	27.9688	27.9302	27.9320
+ **hv**	28.0260	28.0228	27.9302	27.9289
+ **rotate**	28.0365	28.0332	27.9350	27.9320
+ **hv rotate**	**28.0441**	28.0414	27.9403	27.9371

Table 2. Ablation table in pos processing

I have tried adding a sharpening adjustment, but I couldn't find a hyperparameter suitable for all photos, so I excluded it.

2.1 Codalab Result

The following table presents the PSNR score, based on the bicubic down scaled method, and in turn, zooming back in, you can know that the most basic PSNR score should be at least 26.

Type	bicubic	baseline	ours
PSNR	26.0654	27.4162	28.0441

#	SCORE	FILENAME	SUBMISSION DATE	SIZE (BYTES)	STATUS	✔	
1	0.0	310653005_answer3_895_aug12.zip	01/12/2022 10:07:40	62318	Finished		+
2	28.0414	310653005_answer3_1239_aug12.zip	01/12/2022 11:18:07	63582	Finished		+
3	28.0441	310653005_answer3_895_aug12.zip	01/12/2022 11:19:15	63582	Finished		+
4	28.0441	653005_answer.zip	01/12/2022 13:36:41	64825	Finished	✔	+

3 Conclusion

The training data must be made by yourself, and the model is based on RCAN-improved DRLN. Sadly I don't have time to try other models to compare. I mainly focus on testing a certain model. I'm not familiar with GANs, and according to the presentation order in the slides, VDSR will take longer to train the model. I chose a light and fast DRLN for training. While the result is higher than baseline 1 (PSNR), it is still a little far from the top. In the future, I will try GAN for comparison. The photo resolution mentioned in the experiment is too high, resulting in insufficient memory. In the future, I want to use this model as a basis, divide the high resolution into suitable sizes for verification, increase the resolution in blocks, and then stitch back a higher image. High-resolution image.

References

[1] Y. Zhang et al.: Image Super-Resolution Using Very Deep Residual Channel Attention Networks. *ECCV*, 2018.

[2] Saeed, Anwar, and Nick Barnes, Densely Residual Laplacian Super-Resolution. *IEEE*, 2019.

[3] Codalab website: https://codalab.lisn.upsaclay.fr/competitions/622?secret_key=4e06d660-cd84-429c-971b-79d15f78d400