VRDL-Homework4

Name: Kuok-Tong-Ng
Student ID: 309652030

GitHub link: K-T-Ng/VRDL-Homework4 (github.com)

1. Introduction

The goal of this homework is to train a model to reconstruct a high-resolution image from low-resolution input. The training dataset contains 291 high-resolution images. On the other hand, the testing dataset contains 14 low-resolution images, we are asked to upscale the low-resolution test images to the high-resolution image with an upscaling factor 3, like the figure below.

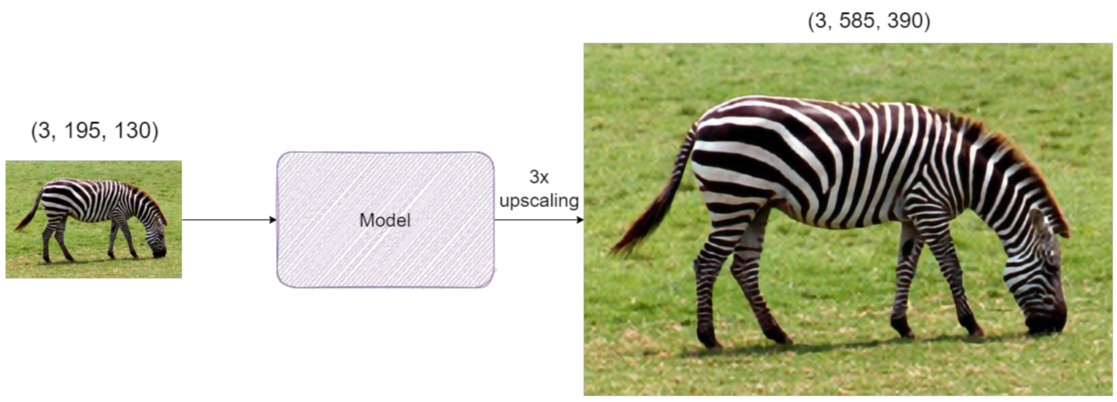


Figure 1 Up sampling by a factor 3.

2. Model Architecture

In this homework, we reference Densely Residual Laplacian Super-Resolution (DRLN) paper [1], which is improve from Residual Channel Attention Network (RCAN) [3]. We try to implement DRLN by using some of the code from [2]. In the following sub-sections, we want to introduce the detail of DRLN and the difference between DRLN and RCAN.

The overall architecture of DRLN and RCAN are similar. As shown in Figure2, DRLN consists of four components: Feature extraction, cascading over residual on the residual, up-sampling and reconstruction. In the following sub-sections, we are going to figure out the detail of DRLN and the difference between DRLN and RCAN.

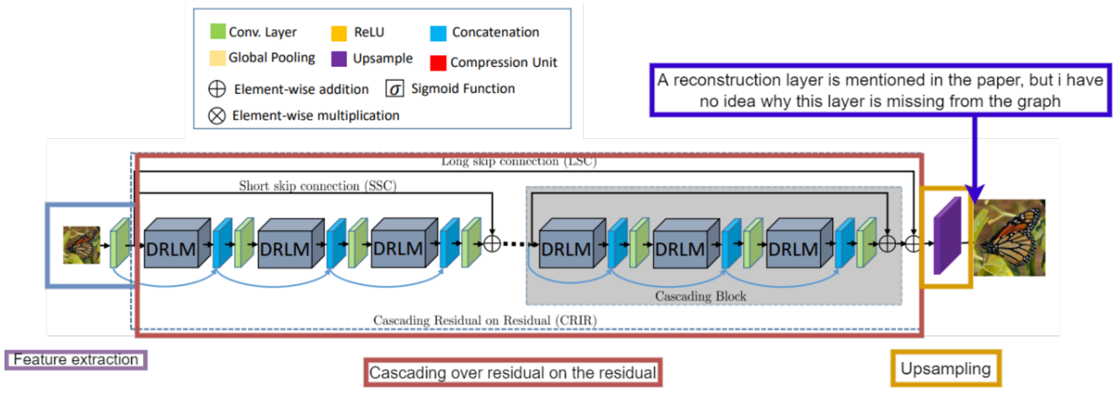

Conv. Layer
ReLU
Concatenation
Global Pooling
Upsample
Compression Unit
Element-wise addition
Sigmoid Function
Element-wise multiplication
A reconstruction layer is mentioned in the paper, but i have no idea why this layer is missing from the graph
Long skip connection (LSC)
Short skip connection (SSC)
DRLM
Cascading Block
Cascading Residual on Residual (CRIR)
Feature extraction
Cascading over residual on the residual
Upsampling

Figure 2 DRLN architecture (From [2])

2.1 Feature Extraction and Reconstruction

This two parts in DRLN and RCAN are the same. Both of them consists only one convolutional layer.

2.2 Cascading Residual on the Residual (CRIR)

Both DRLN and RCAN use a very deep residual in residual (RIR) structure in order to have a large receptive field size, see Figure3. We may see that both of them composed by many blocks (and hence many convolutional layers). Increasing the number of layers also increase the computational cost. RCAN has more than 400 convolutional layers, which needs more computation time. DRLN try to decreasing the number of convolutional layers but with a compatible performance with RCAN. DRLN adopted the method from Dense Net, we will see the details in the following comparisions.

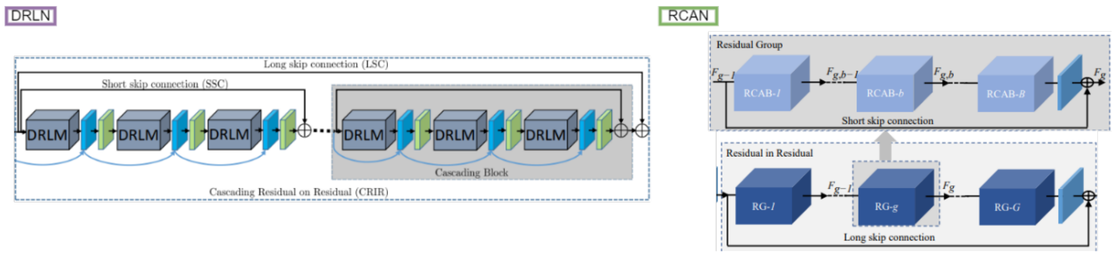


Figure 3 Residual in Residual (RIR) (from [1] and [3])

- **Cascading Block vs Residual Group (RG)**

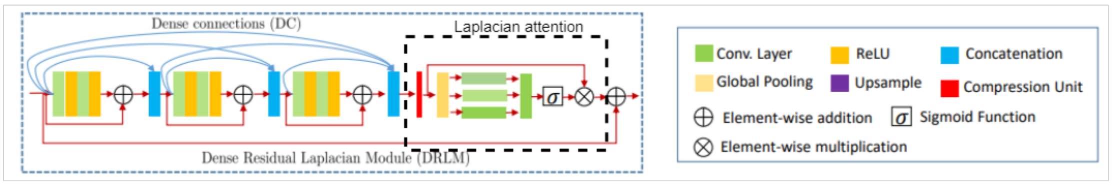
  The first compatible modules in RIR structure is Cascading Block (shaded are in Figure3 left hand side) and Residual Group (RG, the block in Figure3 right hand side), both of them consists many sub-blocks with short skip connection, which make the network pay more attention to high frequency features. The difference in this level is that there are skip-connection between DRLM modules (see the blue line under the DRLM module below in Figure3).

- **DRLM vs RCAB**

  One of the major difference between DRLN and RCAN is Dense Residual

Laplacian Module (DRLM) vs Residual Channel Attention Block (RCAB) since they are the main module in their architecture. The module structure of DRLM and RCAB are shown in Figure4. They both contains attention mechanism, which will introduce later. Their major difference is that, RCAB is a single residual block with attention. DRLM consists several residual block, connected in Dense Net style, then followed by attention. Therefore, DRLN consists skip connections not only between but also within DRLM module. That means every convolutional layer can "reuse" the features that computed before, which can get more information.

DRLM



RCAB

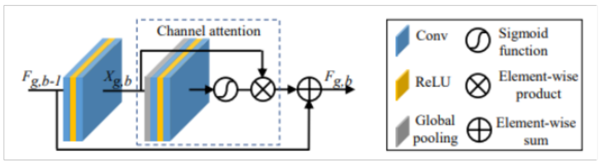


Figure 4 DRLM and RCAB (from [1] and [3])

- Attention Mechanism

Figure5 shows the attention mechanism of DRLN and RCAN. They can be split into five parts.

The first part is global average pooling, this procedure compress the feature map from size $c \times h \times w$ to $c \times 1 \times 1$ by averaging the feature maps in each channel. This can be viewed as a collection of the local descriptors.

Second, Channel-wise downscaling and upscaling try to capture channel-wise dependencies from the aggregated information. In RCAN, both channel-wise downscaling and upscaling are formed by convolution layer, Conv2d with 1x1 kernel, c input channels, c/r output channels. Downscaling reduce the channel size $c \times 1 \times 1$ to $(c/r) \times 1 \times 1$, followed by ReLU activation, Upscaling increase the channel size back to $c \times 1 \times 1$. In DRLN, the only difference is the downscaling process. They adopted three down sampling component. As shown in Figure5, they have used three Conv2d layer with kernel size=3, 3, 3 with padding=3, 5, 7 and dilation=3, 5, 7, respectively. Although they called it Laplacian pyramid attention. I'm still wondering is it different to use a different size of padding

and dilation? Since the input feature map size is just $c \times 1 \times 1$ and the padding are just zeros. The output size is $(c/r) \times 1 \times 1$, and each 3x3 kernel only activate in the middle pixel.
Finally, the output feature vector $c \times 1 \times 1$ is followed by a sigmoid activation function and Element-wise product with the original feature map.

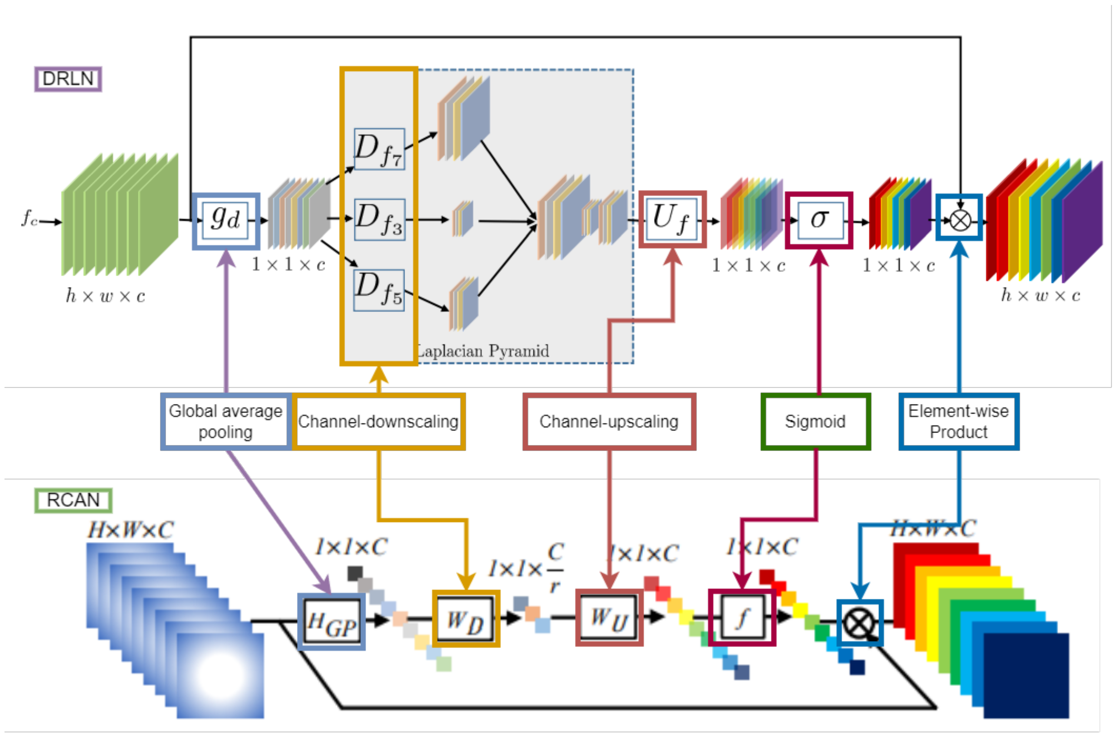


Figure 5 Attention mechanism of DRLN and RCAN (from [1] and [3])

2.3 Up-sampling

For the up-sampling process, both RCAN and DRLN are using Pixel Shuffle [4]. The process can be stated as Figure6. By rearranging a feature map with size $cr^2 \times h \times w$ periodically, we may obtained an output feature map with size $c \times rh \times rw$. For example, if we get a feature map with size $c \times (h/2) \times (w/2)$ (i.e, we want to enlarge the image with a factor 2), we may first apply a convolution layer to increasing the channel number by 4 (i.e, we get $4c \times (h/2) \times (w/2)$). Then apply Pixel Shuffle, then we will get $c \times h \times w$. After up-sampling, we pass the result feature map into the reconstruction layer, which composed by one convolutional layer, has mentioned in section 2.1, then we will get our SR image.

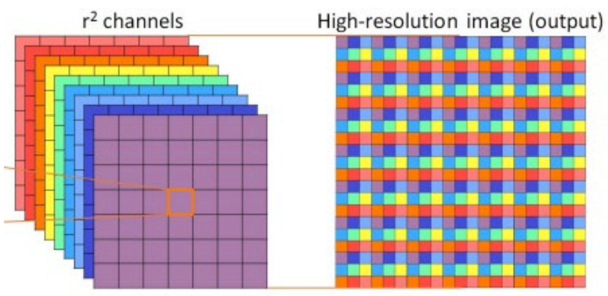


Figure 6 Pixel Shuffle (from [4])

2.4 Performance

DRLN tries to reduce the computational cost by using the connection technique in DenseNet style. Compare to RCAN, which contains more than 400 convolutional layers. DRLN can achieve a compatible performance with mere 160 convolutional layers. Figure7 shows that, although DRLN needs more parameters (since dense connection needs more input channels), it reduce the computation cost very well.

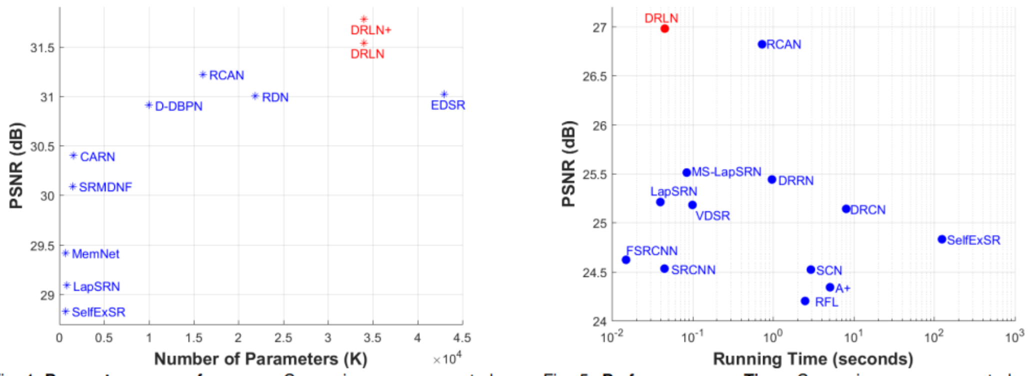


Fig. 4. **Parameters vs. performance.** Comparisons are presented on the MANGA109 [43] for 4× super-resolution.

Fig. 5. **Performance vs. Time.** Comparisons are presented on the URBAN100 [45] for 4× super-resolution. Our proposed method strides a balance between performance and computation time.

Figure 7 DRLN performance (from [1])

3. Methodology

3.1 Model Setting

The model architecture has been introduced in section 2. For the detail, we have used DRLN with six Cascading Blocks (see Figure2). Four of them contains 3 DRLM modules and two of them contains 4 DLRM modules (Therefore, 20 DRLM modules in total).

3.2 Multi-Scale Training

Inspired by VDSR [5] and the code from [2]. I have also tried multi-scale training. By adding multiple up sampling layer with different scale (2x, 3x, 4x) see Figure8. In a particular epoch, we only train a particular scale (For example, in the 15th epoch, we trained for scale 3x, therefore, only Upsample3x layer will be activated, Upsample2x and Upsample4x will not be activated). Moreover, the output is only one SR image, depends on which scale we are training for. This also gives an improvement, we will show in section 4.

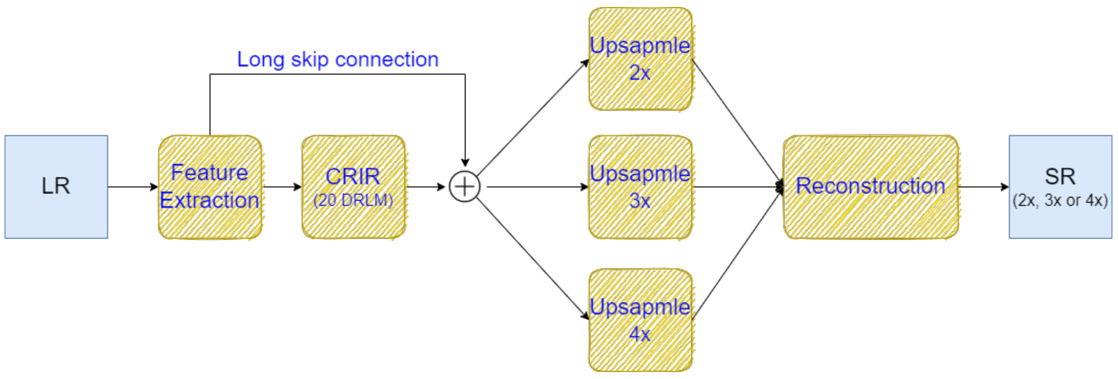


Figure 8 Multi Scale

3.3 Data Pre-Processing

As we have mentioned in section 3.2, we need to prepare three training pairs with different scale (2, 3, 4). We generate the training pairs as follows:

1. Given high resolution image HR and a scale s. we first crop the HR in order its width and height are divisible by s. (For example: HR size with (3, 301, 123) for scale 3, we will crop it into (3, 300, 123))
2. Down scaling with the factor s by using BICUBIC interpolation. Therefore, we get the low-resolution image LR here.
3. (LR, HR) will be a training pair in scale s.

However, the image sizes are variant, before we feed it into the model, we random crop a region from LR image with patch size 48 and crop the corresponding region from HR image. If LR size is less than 48, we will first apply center padding on LR (also in HR), then crop.

3.4 Data Augmentation

Since the patch we cropped in section 3.3 is square, we may apply some augmentation on it. We have applied Identity, horizontal flip, vertical flip, rotate 90 degrees, rotate 180 degrees and rotate 270 degrees with the same probability.

3.5 Hyper Parameters

- ✓ Epochs: 1500
- ✓ Batch size: 16
- ✓ Input patch size: 48
- ✓ Optimizer: Adam
- ✓ Initial LR: 1e-4
- ✓ LR scheduler: LR divide by 2 for every 500 epochs.
- ✓ Loss function: L1-loss

3.6 Self-Ensemble

Self-ensemble is a technique starts from EDSR [6]. In order to increase the psnr in inference. We may apply augmentation during inference. As mentioned in section 3.4, we may apply 6 different augmentation {Identity, horizontal flip, vertical flip, rotate 90 degrees, rotate 180 degrees and rotate 270 degrees} on the LR, and obtain 6 SR result. Then averaging them as the final SR prediction.

4. Summary

In this homework, we try to implement DRLN in order to reconstruct high resolution image from low resolution image. The following shows the result by trying different settings.

Setting	PSNR on testing set
Single scale (3x)	27.3124
Single scale (3x) + augmentation	27.6062
Multi scale + augmentation	28.0268
Multi scale + augmentation + self-ensemble	28.1661

5. Reference

[1] DRLN paper: 1906.12021v2.pdf (arxiv.org)

[2] DRLN code: saeed-anwar/DRLN: Densely Residual Laplacian Super-resolution, IEEE Pattern Analysis and Machine Intelligence (TPAMI), 2020 (github.com)

[3] RCAN paper: 1807.02758.pdf (arxiv.org)

[4] Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network: 1609.05158.pdf (arxiv.org)

[5] VDSR paper: 1511.04587.pdf (arxiv.org)

[6] EDSR paper: 1707.02921.pdf (arxiv.org)