VRDL Homework 4: Super Resolution

Jia-Wei Liao
jw.sc09@nycu.edu.tw
Department of Applied Mathematics, NYCU

February 4, 2023

Abstract

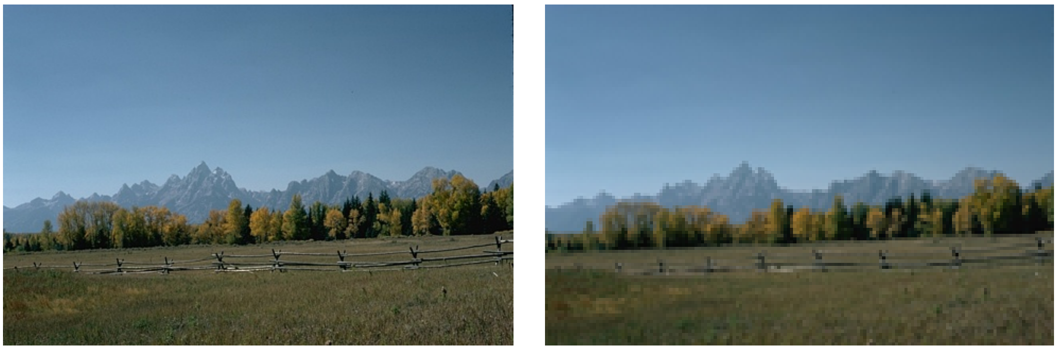
In this homework, we implement deep learning-based techniques to do super-resolution (SR). We use the enhanced Deep Super-Resolution Network (EDSR) [3] and Super-Resolution Feedback Network (SRFBN) [4] which are the famous network for super-resolution in recent years. The metric of the SR task is peak signal-tonoise ratio (PSNR) which is used to quantify reconstruction images. After experimented, we beat the baseline and have the test PSNR of 28.4085 (dB). Our code is available at https://github.com/Jia-Wei-Liao/VRDL.

1 Introduction

In this task, the dataset contains 291 training images with high-resolution (HR) images and 14 test images with low-resolution (LR) images. The test images come from the Set14 dataset. Our task is to restore LR images to HR with 3 scales.

Since we don't have LR of training data, we generate LR training images with 3 scales by bicubic interpolation at the first. Then split the dataset into the training set and validation set. At the inference step, we restore LR images to HR and upload them to the CodaLab [5]. In Fig. 1, we display the HR image and LR image with 3 scales from the training set.

The are many conventional techniques in SR tasks. The simple way is to use bicubic directly. Fast results as no training are required but have a bad result. Moreover, sparse dictionary learning is the machine learning-based method to do this task, but it needs to spend a lot of time calculating. It's not cost-effective. So, in recent years, researchers have begun to study deep learning-based methods to do this work. In Sec. 2.2, we will introduce the EDSR and SRFBN models.



HR image

LR image

Figure 1. Example of training data

2 Methodology

2.1 Data pre-processing

To improve model generalization ability, we use the data augmentation as the following:

- Flip with left and right randomly
- Flip with up and down randomly
- Rotate 90 degrees randomly

2.2 Model architecture

Deep learning-based super-resolution: In 2014, Dong *et al.* proposed the first shallow CNN-based model (SRCNN) [1] on the SR task. Since then, many researchers have developed more CNN-based models and added residual, multiscale, and recurrent mechanisms. In this homework, we implement EDSR and SRFBN models. We will introduce their architecture of them as the following.

2.2.1 Enhanced Deep Super-Resolution Network (EDSR)

Residual learning: EDSR is a CNN-based model with the residual network. In Fig. 2, we can see the residual blocks different with ResNet, SRResNet, and EDSR. There are two differences in this Figure. The first is ResNet has batch normalization but EDSR removes that. The idea is batch normalization layers normalize the feature distribution which makes the feature distribution more concentrated but it may destroy the features of an edge since the features of edge are usually extreme. The second is ResNet has ReLU at the end but SRResNet and EDSR remove that.

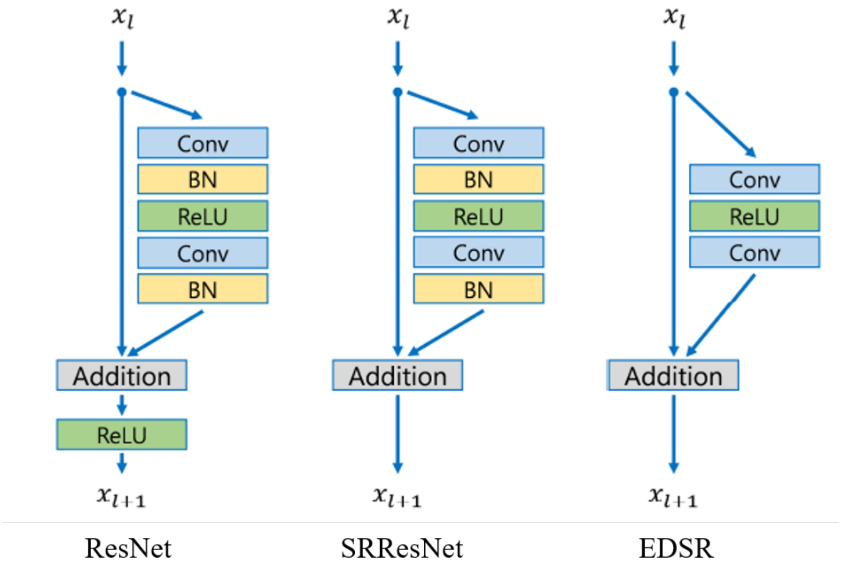


Figure 2. Comparsion of residual blocks

Since EDSR removes the batch normalization layers, it reduces the GPU memory so that we can build a large model deeper and wider and has better performance.

EDSR structure: The architecture of EDSR contains convolution layers, residual blocks, and upsample. It displays in Fig. 3. In particular, the pixel shuffle in the EDSR is origin from ESPCN [2]. In Fig. 4, we shown the ESPCN architecture. The pixel shuffle rearranges the pixels of different channels into a HR image. Since the same pixel on different channels corresponds to the same receptive field, it keeps the same receptive field after pixel shuffle.

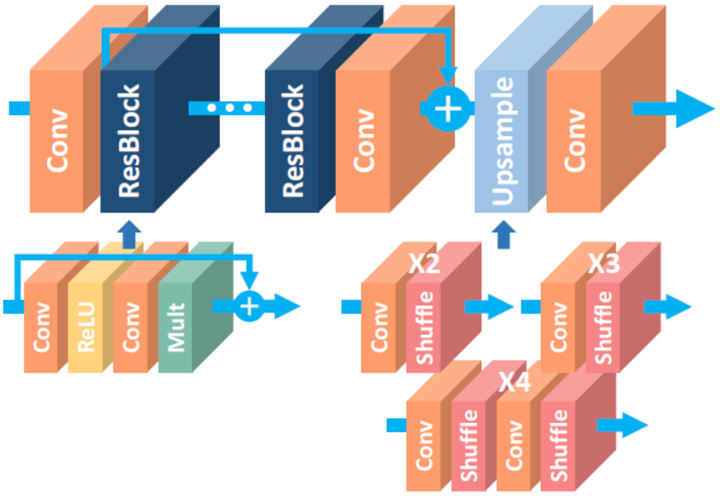


Figure 3. EDSR structure

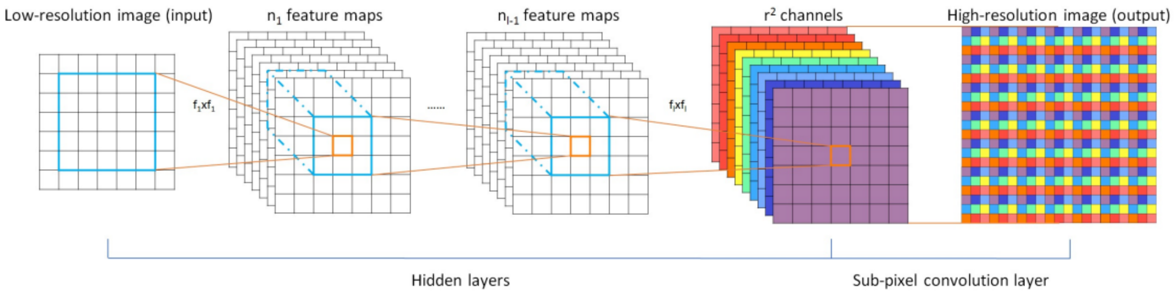


Figure 4. ESPCN structure

2.2.2 Super-Resolution Feedback Network (SRFBN)

Feedback block: The feedback mechanism use recurrent network to enhance image clarity. In Fig. 5, we can see the image become more clearly after many iterations.

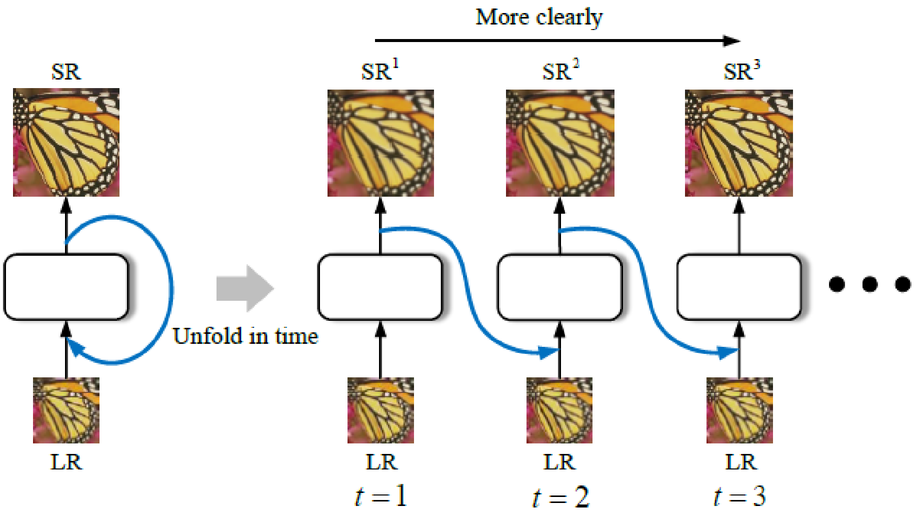


Figure 5. Feedback mechanism

In Fig. 6, we display the feedback block. The feedback block consist of convolution layers, and deconvolution layers. In the following, we introduce the role of each layer in detail.

- **1×1 convolution layer:** refine the channel of concatenation with LR feature map
- **Deconvolution layer:** concatenate all LR images, and then do the up-sample by deconvolution to get HR image
- **Convolution layer:** concatenate all HR images, and then do the down-sample by convolution to get LR image

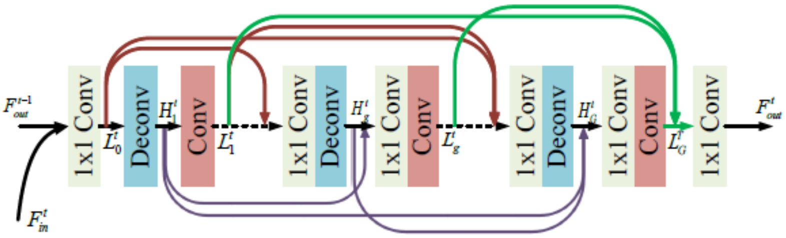


Figure 6. Feedback block

SRFBN structure: The architecture of SRFBN contains convolution layers, feedback blocks, deconvolution layers, and upsample. It displays in Fig. 7. In addition, SRFBN uses PReLU as the activation function which increases the degrees of freedom of the model. In the following, we divide the model architecture into four parts.

1. **Low Resolution Feature Extraction Block (LRFB):** use 3 × 3 convolution layers to extract the feature and then use 1 × 1 convolution layers to revise the channel of output

$$F_{in}^{t} = f_{LRFB}(I_{LR})$$

2. **Feedback Block (FB):** use the feedback mechanism to enhance image clarity.

$$F_{out}^{t} = f_{FB}(F_{out}^{t-1}, F_{in}^{1})$$

3. **Reconstruction Block (RB):** use deconvolution layers and convolution layers to reconstruct the HR image

$$I_{Res}^{t} = f_{RB}(F_{out}^{t})$$

4. **Skip Connection:** Up-sample the input and connect with the output of RB

$$I_{SR}^{t} = I_{Res}^{t} + f_{UP}(I_{LR})$$

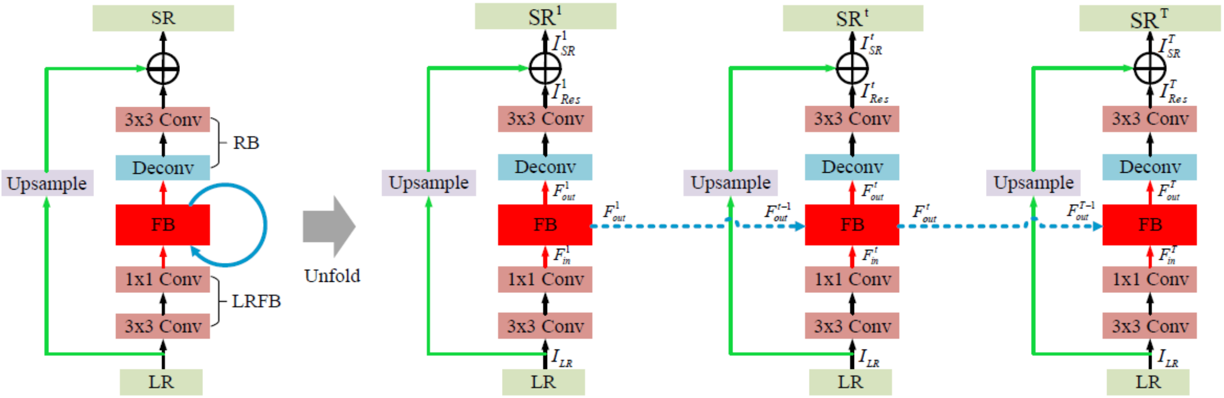


Figure 7. SRFBN structure

2.3 Loss function

In the super-resolution task, it is usually used L1 or L2 loss. For example, SRCNN and VDSR use L2 loss. Since the L2 loss is prone to gradient explosion, the author of VDSR uses gradient clipping to avoid it. On the other hand, EDSR and SRFBN use L1 loss. It's more stable than L2 loss but has poor smoothness. They have their advantages. In the following, we will introduce them in detail.

L1 loss:

$$\ell_1(\hat{y}, y) = |\hat{y} - y|$$

It's defined by absolute function. In Fig. 8, we can see the origin is a sharp point, so we usually set the gradient at the origin to be 0. In addition, the gradient of L1 loss is piecewise constant so that it is less sensitive to different inputs.

L2 loss:

$$\ell_2(\hat{y}, y) = (\hat{y} - y)^2$$

L2 loss is defined by square function. In Fig. 8, we can see the function rises sharply outside 1 and -1, so it is prone to gradient explosion. Nevertheless, when getting closer to the origin, the gradient becomes smoother. It's suitable for optimization between -1 and 1.

In this homework, we tried to replace the loss function with Smooth L1 loss. This inspiration comes from the loss of Faster-RCNN regression. we will see the experiment in Sec. 3

Smooth L1 loss:

$$\ell_S(\hat{y}, y) = \begin{cases} 0.5 \cdot \ell_2(\hat{y}, y) & \text{if } |\hat{y} - y| < 1 \\ \\ \ell_1(\hat{y}, y) - 0.5 & \text{if } |\hat{y} - y| \geq 1 \end{cases}$$

Smooth L1 loss combines the advantages of L1 loss and L2 loss, it not only prevents the gradient explosion, when x is close to 0, the gradient also decreases to 0.

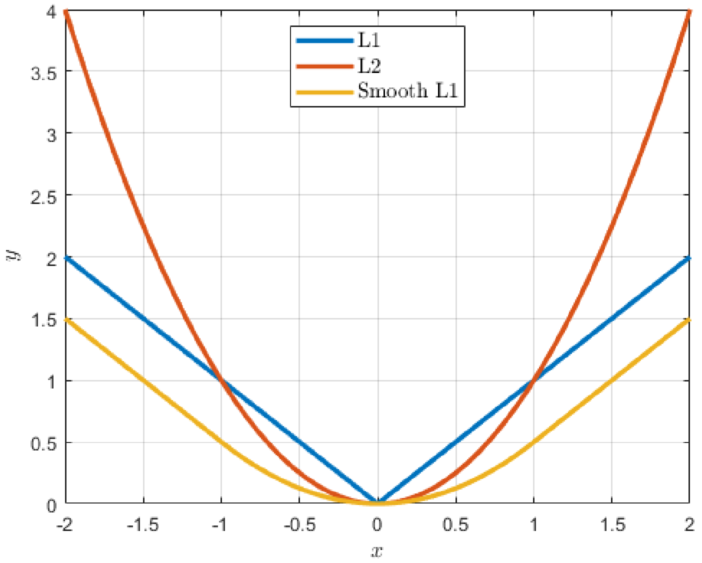


Figure 8. Loss function

Finally, we can compute the total loss by

$$\mathcal{L}(\hat{Y}, Y) = \sum_{i,j,c} \ell_*(\hat{Y}_{i,j,c}, Y_{i,j,c})$$

where ℓ_* is L1, l2, or Smooth L1 loss.

2.4 Optimization

We use Adam and AdamW as the optimizer. AdamW is a stochastic optimization method that modifies the typical implementation of weight decay in Adam, by decoupling weight decay from the gradient update. We give the algorithm like the following:

Algorithm AdamW

```
Input: $f(\theta)$ (objective), $\gamma$ (lr), $\beta_1, \beta_2, \theta_0, \varepsilon, \lambda$ (weight decay)
Initial: $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment),
for $t = 1$ to ... do
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
    $\hat{m}_t \leftarrow \dfrac{m_t}{1 - \beta_1^t}$
    $\hat{v}_t \leftarrow \dfrac{v_t}{1 - \beta_2^t}$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}$
end for
return $\theta_t$
```

To further improve results and make the model converge to the global minimum, we adjust the learning rate by exponent decay at the specific epoch.

2.5 Evaluation metric

We use peak signal-to-noise ratio (PSNR) as the evaluation metric in SR task. PSNR is used to quantify reconstruction images. It's defined as

$$\text{PSNR}(I_{SR}, I_{HR}) = 10\log_{10}\left(\frac{\max(I_{HR})}{\text{MSE}(I_{SR}, I_{HR})}\right)$$

where I_{HR} be a HR image and I_{SR} be a SR image. Since MSE is in the denominator of log, the higher value is the better.

2.6 Self-Ensemble

The author of EDSR adopt the self-ensemble strategy by single model. We represent the formula as the following:

$$I_{SR} = \frac{1}{N}\left[f(I_{LR}) + \sum_{k=1}^{N-1}(T_k^{-1} \circ f \circ T_k)(I_{LR})\right]$$

where I_{LR} be the low resolution image, I_{SR} be the super-resolution image after self-ensemble, f is SR model and $\{T\}_{k=1}^{N-1}$ be the $N-1$ transformations.

3 Experiments

We set the hyper-parameter as the following:

- Batch size: 16
- Epoch: 1500
- Base learning rate: 0.0001
- Learning rate decay factor: 0.5
- Learning rate decay period: 200

First, we split the dataset into 8:2 training set and validation set, but the performance seems not good. So, we increase the training number and only keep 5 images as the validation set. We shown our experiments result in Tab. 1. We can see SRFBN is better than EDSR, and Smooth L1 loss is better than L1 loss and L2 loss. Moreover, self-ensemble can improve the PSNR by 0.02 to 0.07 dB.

model	ratio (train/val)	features	loss	PSNR (dB)	self-ensemble
Bicubic	×	×	×	26.0654	×
EDSR	232/59	32	L1	27.5843	27.6396 (+0.0553)
EDSR	232/59	256	L1	27.9532	27.9745 (+0.0213)
EDSR	286/5	256	L1	28.0577	28.0968 (+0.0391)
SRFBN	232/59	32	L1	27.6197	27.6860 (+0.0662)
SRFBN	232/59	64	L1	28.0728	28.1385 (+0.0657)
SRFBN	286/5	64	L1	28.1393	28.2127 (+0.0734)
SRFBN	286/5	128	L1	28.2680	28.3339 (+0.0659)
SRFBN	286/5	128	L2	28.2253	28.2787 (+0.0534)
SRFBN	286/5	128	Smooth L1	28.2769	28.3409 (+0.0640)
SRFBN	286/5	256	Smooth L1	28.3324	28.4085 (+0.0761)

Table 1. PSNR of test result

In Fig. 9, we show the convergence of EDSR and SRFBN. We can see SRFBN convergent fast and be stable. EDSR was less stable in the previous epoch. It stabilized after about 400 epochs. The PSNR of SRFBN is always higher than EDSR.

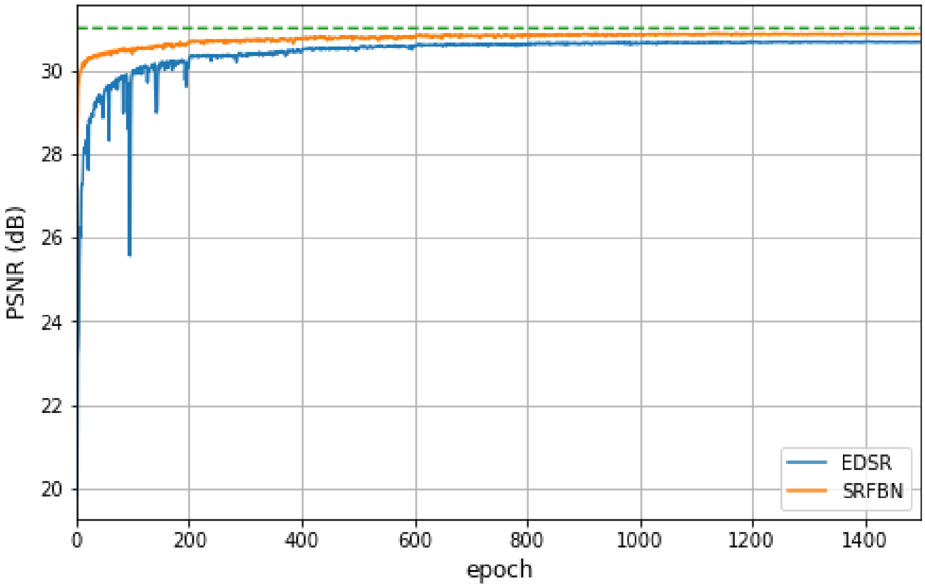
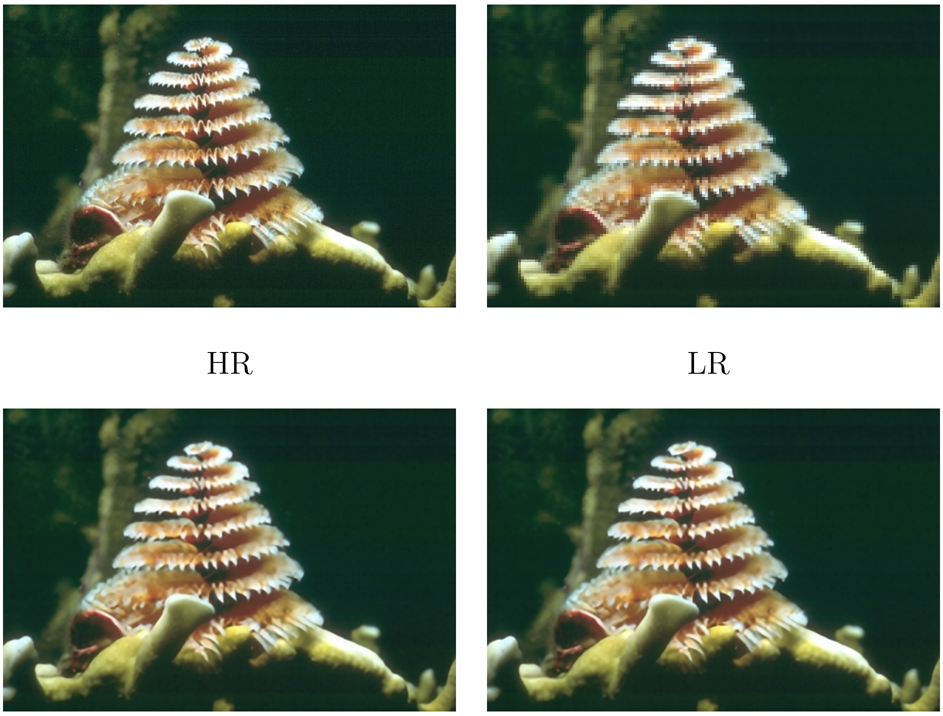


Figure 9. Convergence of EDSR and SRFBN

In Fig. 10, we display the prediction of EDSR and SRFBN from the validation image. The LR image on the top right is rather blurry. The SR images recovered pretty well but some boundaries of objects are still blurry.



HR

LR

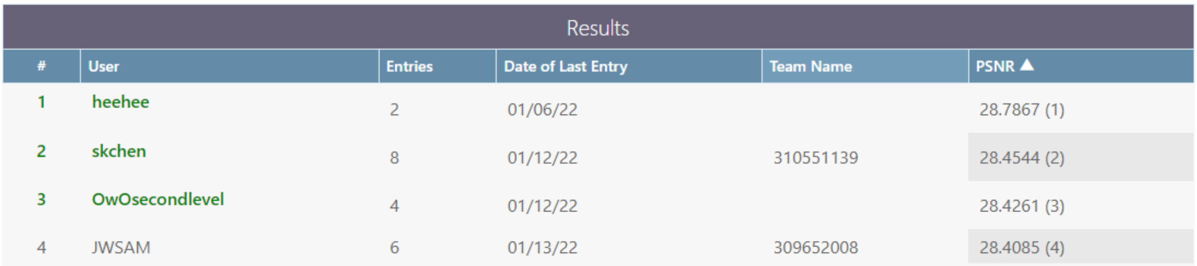
SR (EDSR)

SR (SRFBN)

Figure 10. comparison of EDSR and SRFBN SR

4 Summary

In this homework, we implement EDSR and SRFBN to super-resolution task. EDSR uses the ResNet structure without batch normalization layer and final activation function. It reduces the computation of GPU memory. SRFBN uses a feedback mechanism to enhance image clarity with inexpensive calculations. In addition, We do the data augmentation to improve model generalization ability and try the different number of features and loss functions. We use L1, L2, and Smooth L1 loss. The inspiration for smooth L1 loss comes from Faster-RCNN paper. It's can avoid gradient explosion and be smooth around the origin. After the experiment, we find that SRFBN is better than EDSR. On the other hand, as the feature number increases, the performance gets better. However, due to limited computing resources, we only try feature numbers below 256 but it is enough. In the end, we use self-ensemble as post-processing, it can improve the PSNR by 0.02 to 0.07 dB. Our best score on the Leaderboard is 28.4085 (Fig. 11).



Results

#	User	Entries	Date of Last Entry	Team Name	PSNR ▲
1	heehee	2	01/06/22		28.7867 (1)
2	skchen	8	01/12/22	310551139	28.4544 (2)
3	OwOsecondlevel	4	01/12/22		28.4261 (3)
4	JWSAM	6	01/13/22	309652008	28.4085 (4)

Figure 11. CodaLab

References

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, Image Super-Resolution Using Deep Convolutional Networks, *CVPR*, 2014.

[2] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang, Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network, *CVPR*, 2016.

[3] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee, Enhanced Deep Residual Networks for Single Image Super-Resolution, *CVPR*, 2017.

[4] Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu, Feedback Network for Image Super-Resolution, *CVPR*, 2019. https://github.com/Paper99/SRFBN_CVPR19

[5] CodaLab website:
https://codalab.lisn.upsaclay.fr/competitions/622