MMSR: A Multi-model Super Resolution Framework

Ninghui Yuan, Zhihao Zhu, Xinzhou Wu, Li Shen

MMSR: A Multi-Model Super Resolution Framework

Ninghui Yuan[1], Zhihao Zhu[1], Xinzhou Wu[1], Li Shen[1, *]

[1] School of Computer, National University of Defense Technology
Changsha, Hunan, China, 410073
* Corresponding Author: lishen@nudt.edu.cn

Abstract. *Single image super-resolution (SISR)*, as a fundamental low-level vision problem, has been evaluated in the research community and AI industry. Currently, most methods for reconstructing super-resolution images using deep learning are based on models. Training and reconstruction for different types of images is not differentiated at the model level, which may result in waste of resources. At the same time, because of the particularity of the model training of deep learning, the training process is often atomic, which is a great obstacle to the speed of training. We try to build a framework at the system level to further improve the training speed of the model on the basis of ensuring the reconstruction effect. In this paper, a multi-model super-resolution framework is proposed to choosing training network for different input images according to their characteristics, and all input images are classified by a technique named *TVAT (Total Variance above the Threshold)*. Experimental results indicate that our framework brings a 60% performance speedup on average and exhibits good scalability.

Keywords: Super-resolution, Multi-Model, General Framework, Classification.

1 Introduction

Super-Resolution (SR) technique is used to recover a high resolution image from a single (or a series of) low resolution image(s). This technique has been widely used in the fields including remote sensing, video, medicine and public security, etc. In recent years, with the wide application of deep learning, more and more researches pay their attentions to the study of *single image super resolution (SISR)*.

Since the work of SRCNN [1] given by Dong et al., deep convolution neural work has been the basis of other algorithms of super resolution. This work began the deep-learning-based super resolution algorithms. VDSR [4] is also a revolutionary model in super resolution. Residual blocks are first used in this network. SRGAN [9], proposed by Christian Ledig et al., makes use of GAN [14] in super resolution for the first time. Most of the super resolution models based on deep learning, however, concentrated more on the construction of the deep learning network.

There are still several problems and challenges in current SR techniques. First, the researchers give one model to get reconstruction of all the images, they do not pay much attention to the difference between the images. For example, some images can be smooth, and some images can have more texture. Some image may not need too much

training time to get a good result, so using one model to train all the images can waste some computing resources. Second, the researchers concentrate on the result of the reconstruction image all the time, they do not care about the efficiency. But in certain circumstances, the efficiency is significant as well. For example, there is limit time to train the model and to get the reconstruction image, so the training speed is a significant design objective. Third, there is not a good criterion that can totally fit human eyes. MSE-based criteria can make the reconstruction image too smooth, which does not get a good visual result from human eyes.

In this paper we did a research about the first two challenges. We found that different images in the training dataset have different texture. Some images are smooth and others have fine texture. So the training time of different models differs a lot. Therefore, we propose a *multi-model super resolution (MMSR)* framework. MMSR can choose suitable models for different types of images. With the quality of reconstruction image at a same level, MMSR decrease the training time efficiently. The MMSR framework based on SRGAN [9], with the image training set *DIV_2K*, can reduce 40 percent training time on average. Moreover, the MMSR framework has good scalability. The main contributions of this paper are two-folded:

1) We proposed *MMSR*, a multi-model super-resolution framework. This framework can chose a suitable model according to the characteristics of the input images. Therefore, it can get a balance between training result and training time.
2) We proposed *TVAT (Total Variance above the Threshold)*, a method to classify the training images. This approach can be used to describe the complexity of the image texture, and its does not cause a lot of extra computational overhead. At the same time, since the point where the pixel variation is small has little effect on the image texture calculation, removing these points can also more accurately compare the changes of the larger points.

The rest of this paper is organized as follows. Section 2 lists some related works. Section 3 introduces our MMSR framework and the image classification method in detail. In section 4, experimental results are given and the performance of MMSR is evaluated. And finally in section 5, some conclusions are given.

2 Related works

In recent years, deep learning has been applied to many areas of image processing and analyzing, especially super resolution [1-11]. Reference [1] is a pioneer work that brought super-resolution into deep learning area, in which the authors proposed a simple three-layer convolutional neural network called SRCNN and each layer sequentially deals with feature extraction, non-linear mapping, and reconstruction. The input of SRCNN model needs an extra bicubic interpolation to enlarge the size of image. But this lacks high-frequency information and causes extra computation. Their later work FSRCNN [2] removes the bicubic and inserts a deconvolution layer for reconstruction, which learns an end-to-end mapping. VDSR [4] is also a revolutionary model in the development of super resolution. Residual blocks are first used in this network. Other later works almost use residual blocks in their network. SRGAN [9], proposed by Christian

Ledig et al., makes use of GAN [14] in super resolution for the first time. However, most super resolution algorithms using deep learning focus on the model itself as well as the similarity of the reconstruction image. They did not care more about the efficiency of training, which may cause the waste of computing resources.

In the previous SR research, there was almost no work to make the training speed as an evaluation criterion for the model, but in fact, the training time is one of the most important indicators for a system framework. Training the model that meets the user's shorter training time requirement is one of our goals. At the same time, there is hardly a framework for super resolution in the work of the predecessors. Most of the previous work concentrate on the structure of the deep learning network. In these models, the result of the GAN [14] is significant. Before the GAN network is used to solve the super-resolution problem, the mean square error is often used as a loss function when training the network. Although a high peak signal-to-noise ratio can be obtained in this way, the recovered image usually loses high-frequency details, making people have a not very good visual experience.

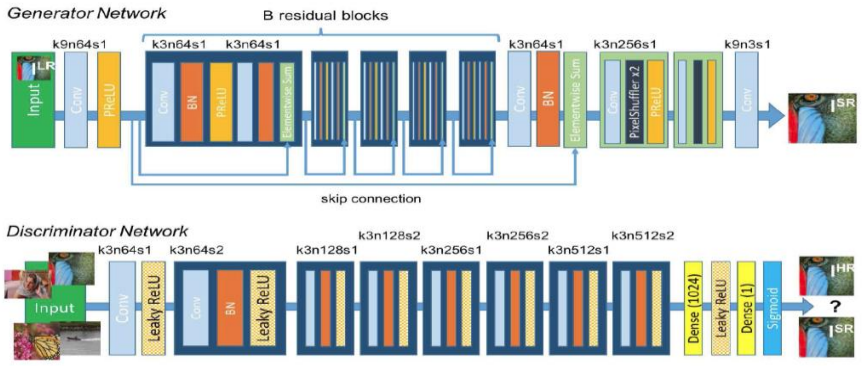

Fig. 1. The structure of SRGAN. The deep learning network currently installed in our framework is this network. We choose different depths for different images, which can greatly improve the efficiency of training. [9]

As shown in Fig. 1, SRGAN [9] uses perceptual loss and adversarial loss to enhance the realism of the recovered image. Perceptual loss is a feature extracted by convolutional neural network. By comparing the features of the generated image after convolutional neural network and the characteristics of the target image after convolutional neural network, the generated image and the target image are semantically and styled more similar. For the current SR problem based on deep learning, the evaluation of the algorithm is mainly divided into two categories, one is in the pixel dimension, such as calculating the MSE, PSNR, SSIM of the image, but this kind of evaluation strategy is easy to make the image to be smooth, fuzzy and other effects. The other type is based on the human eyes' intuitive experience, such as direct observation by the human eye or based on NIQE methods. These methods will guide the image to be sharper, more

intuitive to the human eye, more natural, but also cause more serious distortion. Previous work has improved that there is certain opposite relationship between these two kinds of evaluation methods [13]. We choose to use the perceptual index to be the criteria.

3 MMSR Framework

Our main goal is to build a framework based on SRGAN [9] which is shown in Fig. 1. In this section, we will first introduce our MMSR framework and discuss the performance improvements brought by the framework. Secondly, we will introduce the classification method of image fragments. The MMSR has a training module and a reconstruction module, which are responsible for model training and image combination, respectively. Besides, an image classification module is added to the MMSR framework, and a segmentation layer and a reconstruction layer are added to the reconstruction module. Moreover, the MMSR has good versatility and different deep learning models can be integrated into this framework.

3.1 Frame construction

The first part of MMSR is the training framework, as shown in Fig. 2. The training framework mainly consists of two modules, a classification module and a multi-model training module. The classification module uses the texture features of the image itself to classify the images into different categories. The network model using network models with different parameters, and the classified images *enter* the corresponding model for training and get the corresponding training results.

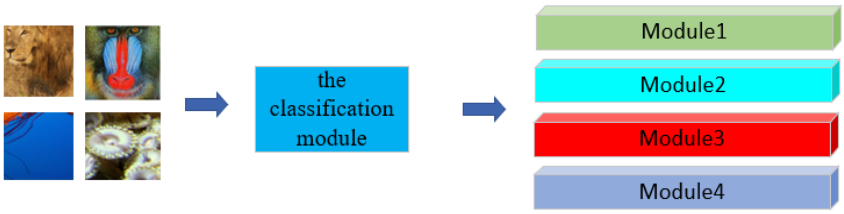

Fig. 2. The structure of our training framework. The training framework consists of two parts: the classification module and the multi-model training module.

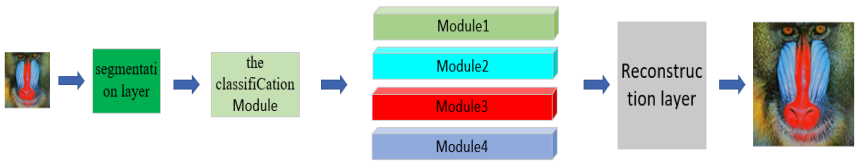

Fig. 3. This reconstruction part is mainly divided into four parts, the segmentation layer, the classification module, the multi-model training module and the reconstruction module.

The second part is a reconstruction part, as shown in Fig.3. This part mainly consists of four parts, a segmentation layer, a classification module, a multi-model training module and a reconstruction module. The segmentation layer is used to cut and block the input low-resolution images that need to be restored. The divided fragments enter the classification module, and the classification module classifies the fragments according to its texture complexity, and the fragments of the classification numbers enter the corresponding model for reconstruction, and finally, the recovered fragments are re-entered into the reconstruction layer for reconstruction and the reconstruction layer assemble the pieces into a complete SR image.

3.2 Image Classification

3.2.1 Overview

Different images have different characteristics. At present, most of the deep learning-based SR methods do not take into account the influence of the nature of the image on the training and recovery process. We consider using the classification module to pre-classify the images, which can make the training and recovery process more targeted, and also improve the efficiency of training. In this MMSR, classification determines the ability of the framework to accelerate to a certain extent. We hope to get a classification method that allows us to train the image to the acceptable effect in the different models, instead of training all the images in the same model.

3.2.2 Total variance above the threshold

We tested some of the images to observe the training rules of different fragments and the features hidden in them. We found that the more complex the texture of the image, the longer the learning time required. Therefore, we try to adopt a suitable way to describe the texture. The simplest description of texture complexity based on the variance of the whole image. For the simple method of calculating the variance of the whole graph, there is a obvious problem that for some grid-like or other images with relatively uniform texture changes, although the variance is relatively small, it seems that the texture is relatively simple, but actually training and reconstruction The difficulty is not small, which will lead to the failure of the classification. We have found that this

method does not work well, so we consider using an innovative method to describe the variance of the pixel variation between each pixel and its neighboring 8 points, as shown in Fig. 4, which we called the variance of single pixel (VSP). And set a threshold. The sum of the values above this threshold is summed to obtain the total variance above the threshold（TVAT）.

The VSP value of the i-th pixel is calculated as follows.

$$VSP_i = \sum_{j=1}^{8}\big((Ri, Gi, Bi) - (Rj, Gj, Bj)\big)^2 \tag{1}$$

The TVAT value of the whole fragment is calculated as follows.

$$TVAT = \left(\sum_{i=1}^{n} VSP_i * judge_{index\,i}\right) / (cols * rows) \tag{2}$$

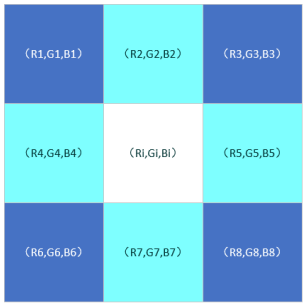

Fig. 4. The variance of the middle grid is determined by the RGB of the surrounding 8 points and its own RGB, and the TVAT of the whole image is determined by the variance of all points.

Among them, *cols* and *rows* respectively represent the number of rows and columns of the fragment, and $judge_{index\,i}$ is a trigger function, which is calculated as follows:

$$judge\_index_i = \begin{cases} 0, & \mathrm{VSP}_i < \mathrm{threshold} \\ 1, & \mathrm{VSP}_i \geq \mathrm{threshold} \end{cases} \tag{3}$$

Table 1. The perceptual index of SRGAN models with different index.

Picture number	TVAT	Perceptual index					
		2-2	4-2	4-6	8-6	16-8	best
1	0.22316	10.7709	10.7093	8.9474	10.6772	13.4531	4-6
2	0.14125	10.7286	10.2412	9.8015	10.6265	11.0304	4-6
3	5.84019	6.8456	6.7848	6.9019	6.8478	6.4810	16-8
4	0.36391	14.9585	10.4451	14.3677	12.9602	11.1268	4-2
5	1.46280	7.0466	6.9128	7.6555	7.4745	7.7626	4-2
6	4.54511	5.8232	5.7170	6.0661	5.4763	5.3578	16-8
7	4.15810	6.4053	6.4270	6.8263	6.4317	6.6969	2-2
8	2.50022	6.8408	6.9172	6.8703	6.7201	7.0015	8-6

9	0.05513	12.8878	9.0682	14.9559	12.1362	14.2192	4-2
10	3.30724	6.6211	6.2182	6.9568	6.1581	6.1914	16-8
11	3.01382	7.2778	6.9914	6.6588	6.6947	6.5318	16-8
12	0.06423	11.7144	10.3977	10.3883	11.1998	11.4946	4-6
13	4.39549	6.6559	6.5765	6.7881	6.6619	5.8498	16-8
14	0.27429	11.2172	10.5885	9.8657	12.9558	10.6933	4-6
15	0.32317	9.6611	9.0914	8.961	9.6730	9.5313	4-6
16	5.95909	7.7423	7.5606	7.6745	7.7127	7.1994	16-8

Threshold is the value to help us classify images. And some of the results of the effect test are listed in Table 1. In Table 1, X-Y means the network has X residual blocks in generator and Y layers in discriminator. For example, 16-8 means that the network has 16 residual blocks in generator and 8 layers in discriminator. It can be seen that the larger the TVAT, the more complicated the image, and the deeper the network depth, the better the effect is. We randomly selected 80 image fragments for testing and found the following rules. As shown in Fig. 5, it can be seen that when the value of TVAT is relatively small, 4-2 and 4-6 have the best effect. When the value of TVAT is between 2 and 4, 2-2 and 16-8 have better effects. When the value of TVAT is large, 4-2 and 16-8 work best, so we can classify images according to TVAT.

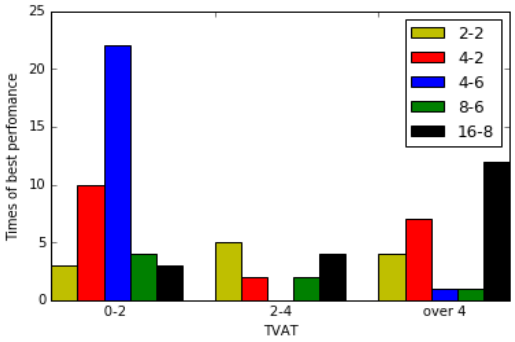

Fig. 5. The usage of TVAT.

3.3 Multi-model training module

After classifying the images, we need to use different models to train the classified images (shown in Fig. 6). We configure different training models on different GPU nodes. These models can be completely different kinds of deep learning network models, or they can be the same. A deep learning network model, but the depth of the network is different. The second way MMSR is currently using it.

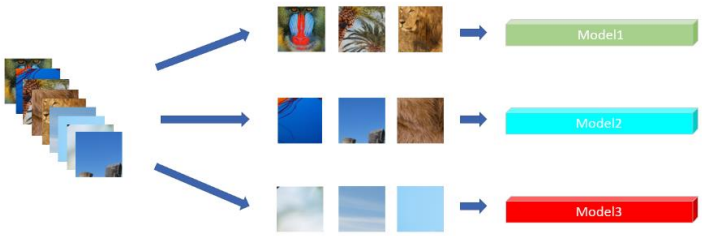

Fig. 6. Model1, model2 and model3 can represent different kinds of network models or the same network models with different depths.

3.4 Reconstruction layer

The reconstruction layer is used to re-splicing the recovered fragments into a complete SR image (shown in Fig. 7). Since different fragments will enter different models for learning after a image is segmented, the recovery time required is different. Need to wait for all the pieces to be recover before reconstruction.

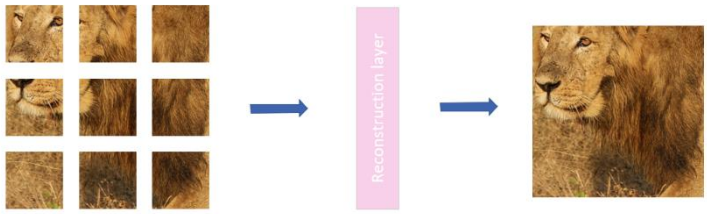

Fig. 7. The different fragments are recovered in different models, and they can reconstruct to a complete SR image through the reconstruction layer.

At the same time, in the process of assembling the fragments into the whole picture, there may be edge effects. As shown in Fig. 8, we leave overlapping parts when cutting, so that in the process of reconstructing, the average value of the overlapping parts can be used to eliminate the edge effect. In our framework, the size of the overlap can be adjusted according to user needs to ensure that users can better eliminate edge effects.

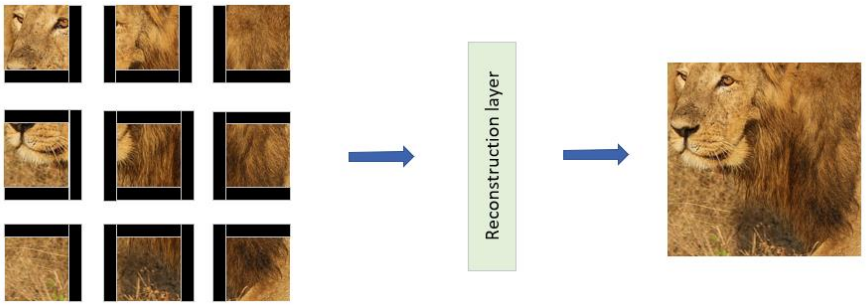

Fig. 8. Overlapping fragments can be combined into a complete SR picture through the reconstruction layer, and edge effects are reduced by averaging the pixel values of the overlapping portions.

4 Experiment Results

4.1 Environment Setup

We construct a cluster which consists of 4 CPU-GPU heterogeneous nodes to evaluate the performance and scalability of our MMSR framework. The main system parameters of each node are listed in Table 2.

Table 2. System parameters of each computation node.

HW/SW Module	Description
CPU	Intel ® Xeon® E5-2660 v3 @2.6GHz x 2
GPU	NVIDIA Tesla K80 x 2
Memory	64 GB
OS	Linux CentOS 7.4
Development Environment	Anaconda 3, Pytorch 1.0

In this work, *DIV_2K* image set is used as both train set and test set. As a widely used image quality metric, the *perceptual index (PI)* value is also used by us to compare different SR frameworks or models. The PI value can be calculated using following formula:

$$Perceptual\ index=12((10-Ma)+NIQE) \tag{4}$$

where *NIQE* (Natural Image Quality Evaluator) is based on the construction of a "quality aware" collection of statistical features based on a simple and successful space domain natural scene statistic (NSS) model. Fig. 8 shows that a lower perceptual index indicates better perceptual quality.

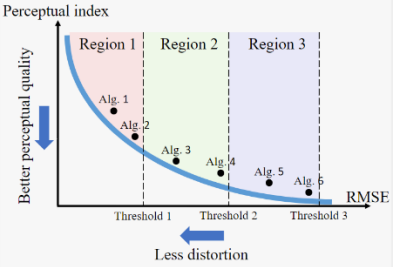

Fig. 9. A lower perceptual index indicates better perceptual quality. We can see that mathematically that distortion and perceptual quality are at odds with each other [12,13].

The advantage of PI value is that different from the traditional image quality evaluating metrics, it can better match the senses of the human eye. Moreover, the GAN model itself tries to improve the sensory level of SR images. Therefore, using PI value as the metric can directly reflect the characteristics of our MMSR framework.

4.2 Experiment details

Since we need to classify the images into different parametric models for learning, we tested the training under different parameters, using the SRGAN model. Firstly, we put the images in SRGAN models with different parameters for training. We observed that different types of images have different training effects under different parameters, and the effect of the model depth training is better. In other words, in a limited training time, the training effect and the depth of the model are not necessarily positively correlated.

According to our framework concept, we use the Python language to implement the framework. The blocker can divide the image into arbitrary blocks as needed. In this experiment, we divide images into three parts according to the TVAT value, and the different parts are sent to different GPU nodes to train. 800 images are trained in each model, and each image we train for 300 times. Finally, the whole image is merged into SR image. Our training time is shortened compared with 16-8 SRGAN [9] (i.e. standard SRGAN), and the effect is better. The training time of these three methods are listed in Table 3.

Table 3. Training time of different methods (database :DIV_2K).

Method	Training time(s)	Average time(s)
SRGAN (one GPU node)	16415.600 17680.502 17417.912	17171.338
MMSR (one GPU node)	10683.279 10592.480 10598.944	10624.901
MMSR	5921.761	5894.845

(three GPU nodes)	5872.322
	5890.451

We use Pytorch framework to train and the test the model. But we do not find a method in Pytorch to train and make training the models with more than one GPU node yet. So we can only train and test the model separately on different GPU nodes.

It can be seen that with the operation result of a GPU node, the acceleration ratio of MMSR is about 1.6. At the same time, we can use multiple GPU nodes for acceleration. At the same time, we can use multiple GPU nodes for acceleration. As we can see, if we use three GPU nodes, the acceleration ratio of MMSR is about 2.9.

Fig. 9 compares the reconstruction effect of MMSR compared with other methods. We can see that the effect of the bicubic interpolation method is relatively poor, and the reconstruction effect of MMSR is not much different from that of SRGAN, and even achieves better results in the reconstruction of some images.

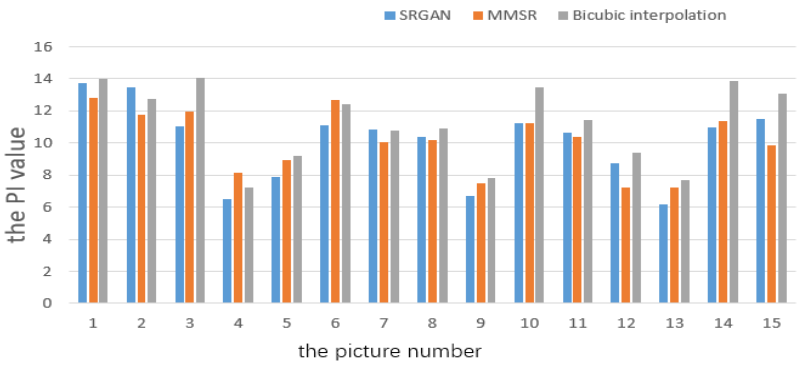

Fig. 10. The reconstruction effect of MMSR compared with SRGAN and Bicubic interpolation. The smaller the value of PI is, the better the visual perception of the result image is.

5 Conclusions

This paper proposes MMSR, a general multi-model framework for super-resolution image reconstruction. The highlight of our work is to build a generic framework to solve the SR series problems at the system level, and create a kind of classification method based on experiments (TVAT) to classify the training set, because we find that in the deep learning network with different parameters, the difference in the training effect of the images with different texture complexity is significant, so the pre-classification can put the same kind of images into the most suitable network. Training in MMSR can greatly improve the efficiency of the training, and at the same time, it will not reduce the training effect too much. In future SR research, no matter how the model progresses, our framework can nest the model into our framework and accelerate its training speed.

References

1. Chao Dong, Chen Change Loy , Kaiming He , Xiaoou Tang: Learning a Deep Convolutional Network for Image Super-Resolution, ECCV (2014).
2. Chao Dong, Chen Change Loy, and Xiaoou Tang： Accelerating the Super-Resolution Convolutional Neural Network，ECCV (2016).
3. Wenzhe Shi , Jose Caballero , Ferenc Huszar´ , Johannes Totz , Andrew P. Aitken , Rob Bishop , Daniel Rueckert , Zehan Wang: Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network, CVPR (2016).
4. Jiwon Kim, Jung Kwon Lee and Kyoung Mu Lee: Accurate Image Super-Resolution Using Very Deep Convolutional Networks, CVPR (2016).
5. Xiao-Jiao Mao, Chunhua Shen, Yu-Bin Yang: Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections, NIPS (2016).
6. Ying Tai, Jian Yang, and Xiaoming Liu: Image Super-Resolution via Deep Recursive Residual Network, CVPR (2017).
7. Wei-Sheng Lai,Jia-Bin Huang ,Narendra Ahuja ,Ming-Hsuan Yang: Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution CVPR (2017).
8. Tong Tong, Gen Li, Xiejie Liu, Qinquan Gao: Image Super-Resolution Using Dense Skip Connections, ICCV (2017).
9. Christian Ledig, Lucas Theis, Ferenc Husz´ar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, CVPR (2017).
10. Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee: Enhanced Deep Residual Networks for Single Image Super-Resolution, CVPR (2017).
11. Xintao Wang, Ke Yu, Chao Dong, Chen Change Loy: Recovering Realistic Texture in Image Super-resolution by Deep Spatial Feature Transform, CVPR (2018).
12. THE PIRM CHALLENGE ON PERCEPTUAL SUPER RESOLUTION, https://www.pirm2018.org/PIRM-SR.html
13. Yochai Blau, Tomer Michaeli: The Perception-Distortion Tradeoff, ECCV (2018).
14. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS), pages 2672–2680, 2014.03