Lightweight and Accurate Recursive Fractal Network for Image Super-Resolution

Juncheng Li
East China Normal University (ECNU)
cvjunchengli@gmail.com

Yiting Yuan
East China Normal University (ECNU)
susanyyt@gmail.com

Kangfu Mei
The Chinese University of Hong Kong, Shenzhen
kangfumei@link.cuhk.edu.cn

Faming Fang*
East China Normal University (ECNU)
fmfang@cs.ecnu.edu.cn

Abstract

Convolutional neural networks have recently achieved great success in image super-resolution (SR). However, we notice an interesting phenomenon that these SR models are getting bigger, deeper, and more complex. Extensive models promote the development of SR, but the effectiveness, reproducibility and practical application prospects of these new models need further verification. In this paper, we propose a lightweight and accurate SR framework, named Super-Resolution Recursive Fractal Network (SRRFN). SRRFN introduces a flexible and diverse fractal module, which enables it to construct infinitely possible topological substructure through a simple component. We also introduce the recursive learning mechanism to maximize the use of model parameters. Extensive experiments show that our SRRFN achieves favorable performance against state-of-the-art methods with fewer parameters and less execution time. All code is available at `https://github.com/MIVRC/SRRFN-PyTorch`.

1. Introduction

Single image super-resolution (SISR) aims to reconstruct a high-resolution (HR) image from its degraded low-resolution (LR) one, which is receiving increasing attention in academia and industry. However, it is still considered as a highly ill-posed problem since the mapping between LR and HR images has multiple solutions.

Recently, convolutional neural networks (CNNs) have shown superior performance in various computer vision tasks, and have greatly promoted the development of SISR. CNN-based SR solutions have become mainstream in the past five years. Among them, Dong et al. propose the SRCNN [6], which is the first SR model that introduces the

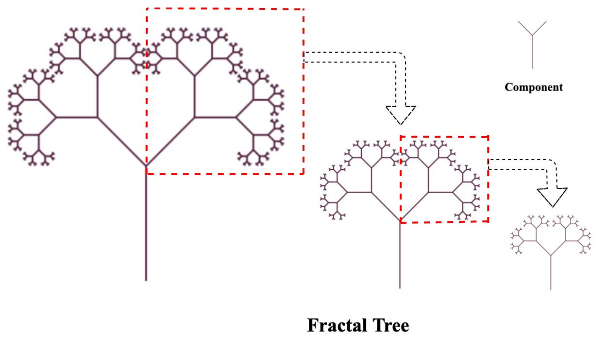

Figure 1. A classical fractal structure: **Fractal Tree**.

CNN to learn the nonlinear LR-to-HR mapping function. SRCNN uses the preprocessed LR image as input, which is upscaled to HR space by some upsampling operators such as Bicubic. However, it will increase the computational complexity and produce visible artifacts. To solve this issue, FSRCNN [7] introduces the deconvolutional layer, and ESPCN [34] proposes an efficient sub-pixel convolutional layer to directly learn the mapping between unprocessed LR and HR images. After that, CNN-based SR methods are blooming and constantly refreshing the best results. Kim et al. propose the VDSR [18] and DRCN [19] by using deeper network, residual learning, skip-connections, or recursive mechanism. LapSRN [20] introduces the Laplacian pyramid structure to progressively reconstruct the sub-band residuals of HR images. In addition, SRResNet [23] introduces residual blocks [12] and EDSR [26] removes batch normalization layers to constructs an intensely wide and deep model. Recently, some models have been proposed to handle SISR via new feature extraction blocks, e.g., Zhang et al. propose the RDN [51] and RCAN [50] by embedding residual learning into the dense block or introducing channel attention mechanism into the residual block, and Li et al.

propose the MSRN [24] by introducing multi-scale residual blocks. All the aforementioned SR models achieve superior performance in terms of PSNR and SSIM [40]. However, SISR has not been completely solved. After investigating plenty of existing SR models, we generalize the following questions and try to answer or solve them in this paper.

(1). The deeper, the better? SRCNN [6] is a 3 layer network, VDSR [18] is a 20 layer network, EDSR [26] is a 65+ layer network, RDN [51] is a 145+ layer network, and RCAN [50] has more than 800 layers. Since VDSR [18] gives the conclusion of "The Deeper, the Better", CNN-based SR models have become deeper and deeper. There is no doubt that increasing the depth of the model is the easiest way to improve the model performance. However, increasing the depth of the network will increase model parameters, execution time, storage space, and memory. In addition, according to our investigation, we found that blindly increasing the depth of the network is not the most sensible way. Each network has its performance bottleneck. When the model reaches the performance bottleneck, it is not profitable to continue to increase the network depth. In other words, if we can make full use of model parameters, the performance of a 100 layer network is almost the same as a 400 layer network. Therefore, we draw a conclusion that making full use of model parameters is more useful than blindly increasing the depth of the model.

(2). Is the channel attention mechanism necessary for SISR? The position attention mechanism [39] and channel attention mechanism [13] (CAM) are the most widely used attention mechanisms in recent years. Among them, the CAM has been widely used in SR and other computer vision tasks [39, 13, 33], e.g., Mei et al. proposes the SrSENet [32] by introducing SEBlock [13] into the SR model, Zhang et al. introduces the CAM [13] into the residual block to construct the residual channel attention block (RCAB [50]), and Dai et al. proposes a second-order channel attention module (SOCAM [4]) for more powerful feature expression. However, does this mean that the CAM is useful? Of course, CAM will bring a slight performance improvement, but the increase in execution time and memory consumption is huge. Taking RCAN [50] as an example, we investigate the impact of CAM on its performance. As shown in Table 1, RIRN is a new model obtained by removing the CAM in RCAN. We can clearly see that the performance of RIRN is dropped by 0.06dB after removing the CAM. However, the execution time is three times faster than RCAN. And this gap will further increase as the input image size increases. Since the improvement is negligible while the cost is huge, we think that the CAM is not necessary for SISR.

(3). Does the previous works on simulating degradation models still meaningful? All the aforementioned models have achieved excellent results in simulating degraded (e.g. Bicubic downsampling) LR images. However, the effect of these models on real images is still not satisfactory. Recently, some new models [8, 42, 43, 48] have been proposed to solve real images SR. We investigate related research and found that these methods often use real images as training dataset or introduce specific learning strategies. Conversely, previous works [6, 26, 51, 50] focus on the design of the network structure, and these structures are often universal. This means that applying real training datasets and related learning strategies to these models will enable them to achieve superior results on real images. Therefore, we believe that efficient network architecture is also crucial.

Method	RIRN($\times 2$)		RCAN($\times 2$) [50]	
Scale	PSNR/SSIM	Time (s)	PSNR/SSIM	Time (s)
Set5	38.24/0.9613	0.21s	**38.26/0.9615**	0.60s
Set14	33.91/0.9206	0.33s	**33.98/0.9210**	1.11s
BSD100	32.37/0.9021	0.24s	**32.39/0.9024**	0.75s
Urban100	33.10/0.9370	1.04s	**33.24/0.9377**	3.78s
Manga109	39.31/0.9784	1.22s	**39.37/0.9785**	4.55s
Average	35.39/0.9399	0.61s	**35.45/0.9402**	2.16s

Table 1. The performance comparison with and without the CAM.

(4). How to design a network with infinite possibilities? More and more well designed network are proposed to solve SISR. However, most of them are sensitive to the subtle network architecture changes and some of them are difficult to reach the level of the original paper. Thence, it is important to build a new network by reusing existing modules that have already been validated. Fractal structures have peculiar structures that can produce infinitely possible results via their self-similarity and infinitely fine structure. Therefore, we aim to use the fractal structure to simplify the model design and provide a more robust model.

To address the above problems, we propose the Super-Resolution Recursive Fractal Network (SRRFN). which achieves superior results with fewer parameters (1/4 of RCAN [50]) and less execution time (1/3 of RCAN [50]). SRRFN removes all unnecessary, time-consuming modules and introduces a new fractal module (FM). The fractal module can generate an unlimited number of topological structures based on a simple component through its unique characteristics. These topologies subnets enable the network to detect rich image features while increasing the fault tolerance of the model. To further improve model performance, we also introduce the recursive strategy to maximize the use of model parameters. In summary, our contributions are:

(i). We propose a fractal module (FM) to simplify the model design, which can generate an infinite number of new structures via a simple component.

(ii). We develop a Super Resolution Recursive Fractal Network, which introduces the fractal module and recursive learning mechanism to maximize the model performance.

(iii). SRRFN achieves superiors results with fewer parameters and faster execution time. Especially, it achieve state-of-the-art results in BD and DN degrade models.

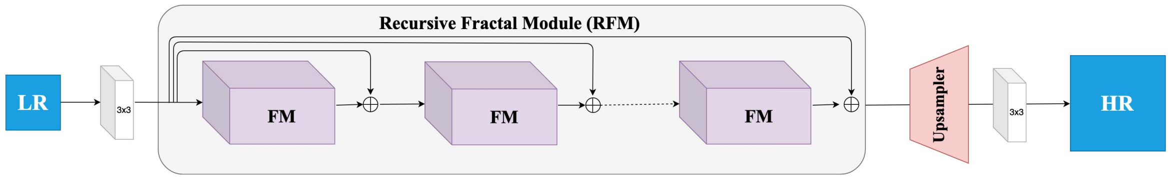

Figure 2. The architecture of our proposed Super-Resolution Recursive Fractal Network (SRRFN). All FMs are weight sharing.

2. Related Work

2.1. Fractal Structure

The fractal structure was proposed by B.B.Mandelbrot in 1973, which is usually defined as “a rough or fragmentary geometry, it can be divided into several parts, and each part is (at least approximately) an overall reduced shape”. It has the following characteristics: (a). self-similarity (b). infinitely fine structure (c). can be defined by a simple method and generated by recursion and iteration. Furthermore, fractal structures are ubiquitous in nature, such as snowflakes, coastlines, leaves, and human organs. Figure 1 is a fractal tree, and each trunk can be viewed as a new tree with similarity to the original tree. This structure enables the creation of an infinite network structure with simple components. At the same time, this structure has been proven to be more fault tolerant, stable, and robust. Therefore, the fractal structure will be the focus of our research.

2.2. Recursive Network

Recursive neural networks are often applied to temporal and sequential data, which have shown superior performance in various computer vision tasks [5, 11, 17, 29]. However, these models are limited on single static image task. To address this drawback, the feedback mechanism is often applied to the recursive network for better learning, which allows the network to carry a notion of output to correct previous states. In SISR, Kim et al. propose a deeply-recursive convolutional network (DRCN [19], up to 16 recursions) for image reconstruction. Tai et al. propose a Deep Recursive Residual Network (DRRN [35]) that introduces local and global residual learning into recursive network. Li et al. propose an image super-resolution feedback network (SRFBN [25]) to refine low-level representations with high-level information and can create the final HR image step by step. However, the recursive-supervision and curriculum learning strategy used in the aforementioned models do not bring significant benefits, but complicate the training process. In this work, we aim to design a simple but efficient recursive network for SISR.

3. SRRFN

As shown in Figure 2, the Super-Resolution Recursive Fractal Network (SRRFN) includes two convolutional layers, a recursive fractal module (RFM), and an upsampling module. The convolutional layer at the head and tail of the network is used to change the image feature dimension. The recursive fractal module (RFM) is composed of a series of fractal modules (FMs). It is worth noting that these FMs are weight sharing and they are essentially the same module. Inspired by the feedback mechanism, we use the output of the current FM as the input of the next FM. We also introduce residual learning in the recursive structure to achieve recursive residual learning.

Define I_{LR} and I_{SR} as the input and output of SRRFN. $L_{in}^{'}$, $L_{out}^{'}$, and L_{sr} denote the input of RFM, the output of RFM, and the output of upsampling module, respectively. Like previous studies, we use a 3×3 convolutional layer to upgrade images to a higher dimension

$$L_{in}^{'} = F_{in}(I_{LR}), \tag{1}$$

where $F_{in}(\cdot)$ is the corresponding operation that converts images from RGB space to higher dimensional space and $L_{in}^{'}$ serves as the input of RFM for feature extraction

$$L_{out}^{'} = F_{RFM}(L_{in}^{'}), \tag{2}$$

where $F_{RFM}(\cdot)$ denotes the recursive fractal module (RFM), which can extract abundant image features from the LR image. Then, all extracted image features are sent to the upsampling module for SR image reconstruction

$$L_{sr} = F_{UP}(L_{out}^{'}), \tag{3}$$

where $F_{UP}(\cdot)$ represents the upsampling module, which uses sub-pixel convolutional layer to learn an array of upscaling filters to upscale image feature maps $L_{out}^{'}$ into the SR output. Finally, a 3×3 conventional layer is applied to L_{sr} to convert it to RGB space and get the final SR image.

$$I_{SR} = F_{out}(L_{sr}). \tag{4}$$

During training, SRRFN is optimized with $L1$ loss function. Given a training dataset $\left\{I_{LR}^{i}, I_{HR}^{i}\right\}_{i=1}^{N}$, we solve

$$\hat{\theta} = arg\ \min_{\theta} \frac{1}{N}\sum_{i=1}^{N}\left\|F_{\theta}(I_{LR}^{i}) - I_{HR}^{i}\right\|_{1}, \tag{5}$$

where θ denotes the parameter set of our model and $F(\cdot)$ represents our SRRFN. Each module of the network will be described in the following sections and the training process will be introduced in the Sec.4.

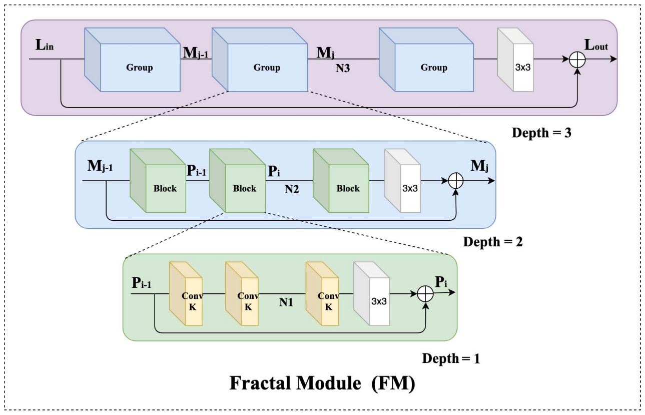

Figure 3. The architecture of our proposed Fractal Modul (FM).

3.1. Fractal Module (FM)

As described in Sec.2.1, the self-similarity and infinite fine structure of the fractal structure make it possible to create an infinite variety of new structures through a simple component. Inspired by this, we combine the fractal structure with CNN to propose the fractal module (FM).

The FM has no fixed structure. It consists of a basic component and a parameter (fractal depth, D). Different network structures can be generated by changing the component and depth. In order to better explain the working mechanism of the module, we provide a specific case in Figure 3. In Figure 3, the purple area denotes the fractal module (FM), which contains N_3 Goups (blue box) and a 3×3 convolutional layer. Each goup contains N_2 Blocks (green box) and a 3×3 convolutional layer. Similarly, each block contains N_1 Convs (Convolutional layer, yellow box) and a 3×3 convolutional layer. In this case, we set $D = 3$ and introduce residual learning in each stage. Thus, this FM can also be considered as a residual fractal network. The 3×3 convolutional layer applied in the tail of each stage is used to achieve local feature fusion, which can further improve the model performance. It is worth noting that we introduce three new parameters (N_1, N_2, and N_3) in this case, which is used to control the number of feature extraction blocks in each stage to make the model structure more diverse. When $N_1 = N_2 = N_3$, it is a standard fractal network. When $N_1 \neq N_2 \neq N_3$, it can still be seen as a fractal network since statistical self-similarity is considered as a special case of self-similarity.

The basic component of this FM is setting as: $D = 1, N_1 = 1, K = 3$ (K is the kernel size). We also introduce residual learning in this component. Thus, this component can be seen as a simple residual block. Compared with the standard residual block [12] and residual channel attention block [50], we remove batch normalization layer and channel attention layer, respectively. This makes our model more flexible and efficient. The operation of this case can be defined as

$$P_i = C^{d1}_{3\times3}(C_{N_1}(\ldots(C_1(P_{i-1}))\ldots)) + P_{i-1}, \quad (6)$$

$$M_j = C^{d2}_{3\times3}(B_{N_2}(\ldots(B_1(M_{j-1}))\ldots)) + M_{j-1}, \quad (7)$$

$$L_{out} = C^{d3}_{3\times3}(G_{N_3}(\ldots(G_1(L_{in}))\ldots)) + L_{in}, \quad (8)$$

where P_i is the output of the i-th block, M_j is the output of the j-th group, and L_{out} is the output of the fractal module. $C(\cdot)$, $B(\cdot)$, and $G(\cdot)$ denote the operation of the conventional layer, the residual block, and the residual group in different depth, respectively.

The fractal module (FM) introduces the fractal structure into CNN, enabling it to construct an infinite variety of topologies structure with a simple component. These topological subnets generate a large number of search paths, enabling the model to detect rich image features while improving the model's fault tolerance and robustness.

3.2. Recursive Mechanism (RM)

In Sec.2.2, we introduced the benefits of recursive networks. In this paper, we aim to explore a lightweight and accurate SR model. To further reuse model parameters, we combine our fractal module with recursive network. As shown in Figure 2, the gray block is the unfolded structure of the recursive fractal module (RFM). In the different recursive stage, FM is weight sharing. In other words, there is only one FM in the RFM, and it appears multiple times in different recursive stages. By introducing the feedback mechanism, the results of the current stage are served as the inputs to the next stage. We also adopt residual learning in the recursive block to achieve recursive residual learning. This allows multiple paths between the input and output of our fractal module, which can learn highly complex features and increase the utilization of model parameters. Therefore, we formulate our RFM as

$$L^s = F_{FM}(L^{s-1}) + L^0, \quad (9)$$

where $s = 1, 2, 3, \cdot\cdot\cdot, S$ (S is the number of recursive stages) and $F_{FM}(\cdot)$ denotes the operation of the fractal module. L^{s-1} and L^s are the input and output of the s-th recursive stages, respectively.

3.3. Integration with Modern Modules

Various neural network modules have been proposed in recent years, including but not limited to residual block [12], dense block [14], memory block [36], residual dense block [51], and res2net block [9]. All of them can be used as the component of our fractal module. This means that by combining existing models with our fractal framework, plenty of new powerful networks can be created. Meanwhile, we can easily integrate the dilated convolutions [44] and shuffleNet mechanism [49, 28] with the proposed fractal module to reduce the model parameters.

Algorithm	Scale	Set5 [3] PSNR / SSIM	Set14 [45] PSNR / SSIM	BSDS100 [2] PSNR / SSIM	Urban100 [15] PSNR / SSIM	Manga109 [31] PSNR / SSIM	Average PSNR / SSIM
Bicubic	×2	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	26.88 / 0.8403	30.80 / 0.9339	30.23 / 0.8832
SRCNN [6]	×2	36.66 / 0.9542	32.45 / 0.9067	31.36 / 0.8879	29.50 / 0.8946	35.60 / 0.9663	33.11 / 0.9219
LapSRN [21]	×2	37.52 / 0.9591	33.08 / 0.9130	31.80 / 0.8950	30.41 / 0.9101	37.27 / 0.9740	34.02 / 0.9302
VDSR [18]	×2	37.53 / 0.9590	33.05 / 0.9130	31.90 / 0.8960	30.77 / 0.9140	37.22 / 0.9750	34.09 / 0.9314
MemNet [35]	×2	37.78 / 0.9597	33.28 / 0.9142	32.08 / 0.8978	31.31 / 0.9195	37.72 / 0.9740	34.43 / 0.9330
SRMDNF [47]	×2	37.79 / 0.9601	33.32 / 0.9159	32.05 / 0.8985	31.33 / 0.9204	38.07 / 0.9761	34.51 / 0.9342
MSRN [24]	×2	38.07 / 0.9608	33.68 / 0.9184	32.22 / 0.9002	32.32 / 0.9304	38.64 / 0.9771	34.99 / 0.9374
D_DBPN []	×2	38.09 / 0.9600	33.85 / 0.9190	32.27 / 0.9000	32.55 / 0.9324	38.89 / 0.9775	35.13 / 0.9378
MDSR [26]	×2	38.11 / 0.9602	33.85 / 0.9198	32.29 / 0.9007	32.84 / 0.9347	38.96 / 0.9776	35.21 / 0.9386
EDSR [26]	×2	38.11 / 0.9602	33.92 / 0.9195	32.32 / 0.9013	32.93 / 0.9351	39.10 / 0.9773	35.27 / 0.9387
RDN [51]	×2	**38.24 / 0.9614**	34.01 / 0.9212	32.34 / 0.9017	32.89 / 0.9353	39.18 / 0.9780	35.33 / 0.9395
SRRFN (Ours)	×2	38.18 / 0.9612	33.97 / 0.9210	32.35 / 0.9018	33.04 / 0.9361	39.23 / 0.9781	35.35 / 0.9396
SRRFN+ (Ours)	×2	**38.24 / 0.9614**	**34.13 / 0.9224**	**32.39 / 0.9023**	**33.24 / 0.9378**	**39.43 / 0.9786**	**33.49 / 0.9405**
Bicubic	×3	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	24.46 / 0.7349	26.95 / 0.8556	27.31 / 0.7943
SRCNN [6]	×3	32.75 / 0.9090	29.30 / 0.8215	28.41 / 0.7863	26.24 / 0.7989	30.48 / 0.9117	29.44 / 0.8455
VDSR [18]	×3	33.67 / 0.9210	29.78 / 0.8320	28.83 / 0.7990	27.14 / 0.8290	32.01 / 0.9340	30.29 / 0.8630
LapSRN [21]	×3	33.82 / 0.9227	29.87 / 0.8320	28.82 / 0.7980	27.07 / 0.8280	32.21 / 0.9350	30.36 / 0.8631
MemNet [37]	×3	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	27.56 / 0.8376	32.51 / 0.9369	30.62 / 0.8669
SRMDNF [47]	×3	34.12 / 0.9254	30.04 / 0.8382	28.97 / 0.8025	27.57 / 0.8398	33.00 / 0.9403	30.74 / 0.8692
MSRN [24]	×3	34.48 / 0.9276	30.40 / 0.8436	29.13 / 0.8061	28.31 / 0.8560	33.56 / 0.9451	31.18 / 0.8757
MDSR [26]	×3	34.66 / 0.9280	30.44 / 0.8452	29.25 / 0.8091	28.79 / 0.8655	34.17 / 0.9472	31.46 / 0.8790
EDSR [26]	×3	34.65 / 0.9280	30.52 / 0.8462	29.25 / 0.8093	28.80 / 0.8653	34.17 / 0.9476	31.48 / 0.8793
RDN [51]	×3	34.71 / 0.9296	30.57 / 0.8468	29.26 / 0.8093	28.80 / 0.8653	34.13 / 0.9484	31.49 / 0.8799
SRRFN (Ours)	×3	34.74 / 0.9296	30.62 / 0.8478	29.29 / 0.8100	28.98 / 0.8689	34.36 / 0.9491	31.60 / 0.8811
SRRFN+ (Ours)	×3	**34.84 / 0.9303**	**30.70 / 0.8490**	**29.35 / 0.8110**	**29.21 / 0.8721**	**34.66 / 0.9505**	**31.75 / 0.8826**
Bicubic	×4	28.42 / 0.8104	26.00 / 0.7027	25.96 / 0.6675	23.14 / 0.6577	24.89 / 0.7866	25.62 / 0.7250
SRCNN [6]	×4	30.48 / 0.8628	27.50 / 0.7513	26.90 / 0.7101	24.52 / 0.7221	27.58 / 0.8555	27.40 / 0.7804
VDSR [18]	×4	31.35 / 0.8830	28.02 / 0.7680	27.29 / 0.7267	25.18 / 0.7540	28.83 / 0.8870	28.13 / 0.8037
LapSRN [21]	×4	31.54 / 0.8850	28.19 / 0.7720	27.32 / 0.7270	25.21 / 0.7560	29.09 / 0.8900	28.27 / 0.8060
MemNet [37]	×4	31.74 / 0.8893	28.26 / 0.7723	27.40 / 0.7281	25.50 / 0.7630	29.42 / 0.8942	28.46 / 0.8094
SRMDNF [47]	×4	31.96 / 0.8925	28.35 / 0.7787	27.49 / 0.7337	25.68 / 0.7731	30.09 / 0.9024	28.71 / 0.8161
MSRN [24]	×4	32.25 / 0.8958	28.63 / 0.7833	27.61 / 0.7377	26.20 / 0.7905	30.57 / 0.9103	29.05 / 0.8235
D_DBPN [10]	×4	32.47 / 0.8980	28.82 / 0.7860	27.72 / 0.7400	26.38 / 0.7946	30.91 / 0.9137	29.26 / 0.8265
RDN [51]	×4	32.47 / 0.8990	28.81 / 0.7871	27.72 / 0.7419	26.61 / 0.8028	31.00 / 0.9151	29.32 / 0.8292
EDSR [26]	×4	32.46 / 0.8968	28.80 / 0.7876	27.71 / 0.7420	26.64 / 0.8033	31.02 / 0.9148	29.33 / 0.8289
MDSR [26]	×4	32.50 / 0.8973	28.72 / 0.7857	27.72 / 0.7418	26.67 / 0.8041	31.11 / 0.9146	29.34 / 0.8287
SRRFN (Ours)	×4	32.56 / 0.8993	28.86 / 0.7882	27.75 / 0.7424	26.78 / 0.8071	31.22 / 0.9159	29.43 / 0.8306
SRRFN+ (Ours)	×4	**32.66 / 0.9006**	**28.95 / 0.7900**	**27.81 / 0.7437**	**26.98 / 0.8113**	**31.56 / 0.9190**	**29.59 / 0.8329**

Table 2. **BI** quantitative comparisons with state-of-the-art SR methods. Best results are **highlighted** and second best results are underlined. Obviously, our SRRFN outperforms all SR methods on all benchmark datasets. The 'Average' represents average results of these 5 datasets.

4. Experiments

Following previous works [24, 26, 50, 51], we only use DIV2K (1-800) [1] as our training dataset. For testing, we choose Set5 [3], Set14 [45], BSDS100 [30], Urban100 [16], and Manga109 [31]. All of them are the widely used test benchmark datasets, which contain a variety of scenarios that can fully validate the model performance.

4.1. Implementation Details

Model setting: In this paper, we propose a general framework for SR, named Super-Resolution Recursive Fractal Network (SRRFN). The core module of SRRFN is the fractal module (FM). As described in Sec.3, the fractal module has no fixed structure. It can be expanded to various networks via changing the component and fractal depth. In order to verify the validity of the model, we provide a simple case. As shown in Figure 3, we use this structure as our fractal module to construct the final SRRFN. We set $D = 3, K = 3, N_1 = 1, N_2 = 10, N_3 = 5$, and $S = 4$ in the final model. We also introduce self-ensemble to improve SRRFN, which is expressed as SRRFN+.

Training setting: During training, we use RGB images as input and augment the training data with flipping horizontally and vertically. Following previous works [24, 26, 50], we randomly extract 16 LR patches with the size of 48×48 as input, and 1,000 iterations of back-propagation constitute an epoch. The learning rate is initialized to 10^{-4} and halved every 200 epochs. We implement our model with the PyTorch framework and update it with Adam optimizer. All our experiments are performed on GTX TitanX.

Degradation Models: In order to demonstrate the effectiveness of our proposed SRRFN, we use three degraded models (**BI, BD**, and **DN**) to obtain LR images. **BI** is the most widely used degraded model to simulate LR images, which is essentially a bicubic downsampling operation that

Algorithm	Scale	Set5 [3] PSNR / SSIM	Set14 [45] PSNR / SSIM	BSDS100 [2] PSNR / SSIM	Urban100 [15] PSNR / SSIM	Manga109 [31] PSNR / SSIM	Average PSNR / SSIM
DRCN [19]	×2	37.63 / 0.9584	33.06 / 0.9108	31.85 / 0.8947	30.76 / 0.9147	37.63 / 0.9723	34.19 / 0.9302
MS-LapSRN [22]	×2	37.78 / 0.9600	33.28 / 0.9150	32.05 / 0.8980	31.15 / 0.9190	37.78 / 0.9760	34.41 / 0.9336
DRRN [35]	×2	37.74 / 0.9590	33.23 / 0.9140	32.05 / 0.8970	31.23 / 0.9190	37.92 / 0.9760	34.43 / 0.9330
SRFBN [25]	×2	38.11 / 0.9609	33.82 / 0.9196	32.29 / 0.9010	32.62 / 0.9328	39.08 / 0.9779	35.18 / 0.9384
SRRFN (Ours)	×2	**38.18 / 0.9612**	**33.97 / 0.9210**	**32.35 / 0.9018**	**33.04 / 0.9361**	**39.23 / 0.9781**	**35.35 / 0.9396**
DRCN [19]	×3	33.85 / 0.9215	29.89 / 0.8317	28.81 / 0.7954	27.16 / 0.8311	32.31 / 0.9328	30.40 / 0.8625
MS-LapSRN [22]	×3	34.06 / 0.9240	29.97 / 0.8360	28.93 / 0.8020	27.47 / 0.8370	32.68 / 0.9390	30.62 / 0.8676
DRRN [35]	×3	34.03 / 0.9240	29.96 / 0.8350	28.95 / 0.8000	27.53 / 0.7640	32.74 / 0.9390	30.64 / 0.8524
SRFBN [25]	×3	34.70 / 0.9292	30.51 / 0.8461	29.24 / 0.8084	28.73 / 0.8641	34.18 / 0.9481	31.47 / 0.8792
SRRFN (Ours)	×3	**34.74 / 0.9296**	**30.62 / 0.8478**	**29.29 / 0.8100**	**28.98 / 0.8689**	**34.36 / 0.9491**	**31.60 / 0.8811**
DRCN [19]	×4	31.56 / 0.8810	28.15 / 0.7627	27.24 / 0.7150	25.15 / 0.7530	28.98 / 0.8816	28.22 / 0.7987
DRRN [35]	×4	31.68 / 0.8888	28.21 / 0.7722	27.38 / 0.7240	25.44 / 0.7640	29.46 / 0.8960	28.43 / 0.8090
MS-LapSRN [22]	×4	31.74 / 0.8890	28.26 / 0.7740	27.43 / 0.7310	25.51 / 0.7680	29.54 / 0.8970	28.50 / 0.8118
SRFBN [25]	×4	32.47 / 0.8983	28.81 / 0.7868	27.72 / 0.7409	26.60 / 0.8015	31.15 / 0.9160	29.35 / 0.8287
SRRFN (Ours)	×4	**32.56 / 0.8993**	**28.86 / 0.7882**	**27.75 / 0.7424**	**26.78 / 0.8071**	**31.22 / 0.9159**	**29.43 / 0.8318**

Table 3. **BI** quantitative comparisons with **recursive models**. Best results are **highlighted**. Obviously, our SRRFN outperforms all recursive SR methods on all benchmark datasets. The 'Average' represents average results of these 5 datasets.

Algorithm	Scale	Parameters	Set5 [3] PSNR / SSIM / Time	Set14 [45] PSNR / SSIM / Time	BSDS100 [2] PSNR / SSIM / Time	Urban100 [15] PSNR / SSIM / Time	Manga109 [31] PSNR / SSIM / Time	Average PSNR / SSIM / Time
RCAN [50]	×2	15.44M	**38.27 / 0.9614** / 0.60s	**34.12 / 0.9216** / 1.11s	**32.41 / 0.9027** / 0.75s	**33.34 / 0.9384** / 3.78s	**39.44 / 0.9786** / 4.55s	**35.52 / 0.9405** / 2.16s
SRRFN (Ours)	×2	4.06M	38.18 / 0.9612 / 0.21s	33.97 / 0.9210 / 0.35s	32.35 / 0.9018 / 0.24s	33.04 / 0.9361 / 1.07s	39.23 / 0.9781 / 1.25s	35.35 / 0.9396 / 0.61s
RCAN [50]	×3	15.63M	**34.74 / 0.9299** / 0.34s	**30.65 / 0.8482** / 0.55s	**29.32 / 0.8111** / 0.41s	**29.09 / 0.8702** / 1.89s	**34.44 / 0.9499** / 2.33s	**31.65 / 0.8818** / 1.10s
SRRFN (Ours)	×3	4.24M	**34.74** / 0.9296 / 0.17s	30.62 / 0.8478 / 0.23s	29.29 / 0.8100 / 0.16s	28.98 / 0.8689 / 0.62s	34.36 / 0.9491 / 0.79s	31.60 / 0.8811 / 0.39s
RCAN [50]	×4	15.59M	**32.63 / 0.9002** / 0.30s	**28.87 / 0.7889** / 0.40s	**27.77 / 0.7436** / 0.30s	**26.82 / 0.8087** / 1.21s	**31.22 / 0.9173** / 1.50s	**29.46 / 0.8317** / 0.74s
SRRFN (Ours)	×4	4.21M	32.56 / 0.8993 / 0.16s	28.86 / 0.7882 / 0.19s	27.75 / 0.7424 / 0.16s	26.78 / 0.8071 / 0.47s	**31.22** / 0.9159 / 0.58s	29.43 / 0.8318 / 0.31s

Table 4. Quantitative comparisons (PSNR/SSIM, Parameters, and Execution time) with RCAN [50].

adopting the Matlab function $imresize$ with the option of $bicubic$. To verify the performance of SRRFN in complex scenarios, we produce LR images in more challenging ways (**BD** and **DN**) like [46, 51]. For **BD**, we blur HR images by a Gaussian kernel of size 7×7 with standard deviation 1.6 and downsample the blurred image with scaling factor ×3. To obtain **DN** model LR images, we perform bicubic downsampling on HR images with scaling factor ×3, and then add Gaussian noise with noise $level = 30$.

4.2. Comparisons with state-of-the-arts

We compare SRRFN with more than 20 SR methods to fully verify the model effectiveness, including Bicubic, SCN[41], SRCNN [6], SelfExSR [16], A+ [38], ESPCN [34], FSRCNN[7], LapSRN [21], MS-LapSRN [22], VDSR [18], DRCN [19], DRRN [35], MemNet [35], SRMDNF [47], D_DBPN [10], SRFBN [25], MDSR [26], EDSR [26], RDN [51], MSRN [24], and RCAN [50]. All of SR results are evaluated with PSNR and SSIM on the Y channel of the transformed YCbCr space. Due to page limitations, only a part of the methods are presented.

Results with BI Degradation Model: In Table 2, we show the quantitative comparisons with some advanced SR models, all of them have well-designed structure. In Table 3, we show the quantitative comparisons with some recursive models, including SRFBN [25]. Among them, best results are **highlight** and the **'Average'** denotes average results of these 5 test datasets. Obviously, our SRRFN achieves superior results on all datasets and achieves the best average results on all upsampling factors. Considering that RCAN [50] and SRRFN (Figure 3 case) are structurally similar, we make a detailed comparison of them in Table 4. We can observe that RCAN [50] is slightly better than SRRFN (×2: 0.17dB, ×3: 0.05dB, and ×4: 0.03dB). However, this gap gradually decreases as the upsampling factor increases. It is worth noting that the parameter quantity of SRRFN is about 1/4 of RCAN, and the execution time is 3 times faster than RCAN (time is tested on one GTX TitanX). This means that SRRFN can achieve similar results as RCAN with fewer parameters and less execution time. In Figure 4, we show the visual comparison on ×2, ×3, and ×4. We can clearly see that most SR methods cannot recover clear and right image edges. In contrast, SRRFN can reconstruct more realistic high-frequency details. Compared with RCAN, SR images reconstructed by SRRFN also achieve great visual performance. This again verifies that SRRFN achieves the best balance between model performance, model size, and execution runtime.

Results with BD and DN Degradation Models: We also show SR results in **BD** and **DN** degradation models in Table 5. Following [25, 50], our SRRFN is compared with Bicubic, SRCNN [6], VDSR [18], SRMD(NF) [47],

Model	Methods	Set5 PSNR / SSIM	Set14 PSNR / SSIM	BSDS100 PSNR / SSIM	Urban100 PSNR / SSIM	Manga109 PSNR / SSIM	Average PSNR / SSIM
BD	Bicubic	28.34 / 0.8161	26.12 / 0.7106	26.02 / 0.6733	23.20 / 0.6601	25.03 / 0.7987	25.74 / 0.7318
	SRCNN [6]	31.63 / 0.8888	28.52 / 0.7924	27.76 / 0.7526	25.31 / 0.7612	28.79 / 0.8851	28.40 / 0.8159
	VDSR [18]	33.30 / 0.9159	29.67 / 0.8269	28.63 / 0.7903	26.75 / 0.8145	31.66 / 0.9260	30.00 / 0.8547
	SRMD(NF) [47]	34.09 / 0.9242	30.11 / 0.8364	28.98 / 0.8009	27.50 / 0.8370	32.97 / 0.9391	30.73 / 0.8675
	RDN [51]	34.57 / 0.9280	30.53 / 0.8447	29.23 / 0.8079	28.46 / 0.8581	33.97 / 0.9465	31.35 / 0.8770
	SRFBN [25]	34.66 / 0.9283	30.48 / 0.8439	29.21 / 0.8069	28.48 / 0.8581	34.07 / 0.9466	31.38 / 0.8768
	RCAN [50]	34.70 / 0.9288	30.63 / 0.8462	29.32 / 0.8093	28.81 / 0.8647	34.38 / 0.9483	31.57 / 0.8795
	SRRFN (Ours)	34.77 / 0.9293	30.67 / 0.8469	29.31 / 0.8096	28.85 / 0.8653	34.51 / 0.9489	31.62 / 0.8800
	SRRFN+ (Ours)	**34.86 / 0.9299**	**30.76 / 0.8479**	**29.36 / 0.8105**	**29.06 / 0.8682**	**34.80 / 0.9502**	**31.77 / 0.8813**
DN	Bicubic	24.14 / 0.5445	23.14 / 0.4828	22.94 / 0.4461	21.63 / 0.4701	23.08 / 0.5448	22.99 / 0.4977
	SRCNN [6]	27.16 / 0.7672	25.49 / 0.6580	25.11 / 0.6151	23.32 / 0.6500	25.78 / 0.7889	25.37 / 0.6958
	VDSR [18]	27.72 / 0.7872	25.92 / 0.6786	25.52 / 0.6345	23.83 / 0.6797	26.41 / 0.8130	25.88 / 0.7186
	SRMD(NF) [47]	27.74 / 0.8026	26.13 / 0.6974	25.64 / 0.6495	24.28 / 0.7092	26.72 / 0.8424	26.10 / 0.7402
	RDN [51]	28.46 / 0.8151	26.60 / 0.7101	25.93 / 0.6573	24.92 / 0.7362	28.00 / 0.8590	26.78 / 0.7555
	SRFBN [25]	28.53 / 0.8182	26.60 / 0.7144	25.95 / 0.6625	24.99 / 0.7424	28.02 / 0.8618	26.82 / 0.7599
	SRRFN (Ours)	28.57 / 0.8194	26.69 / 0.7155	25.98 / 0.6630	25.21 / 0.7506	28.21 / 0.8646	26.93 / 0.7626
	SRRFN+ (Ours)	**28.66 / 0.8211**	**26.75 / 0.7169**	**26.02 / 0.6639**	**25.34 / 0.7538**	**28.37 / 0.8672**	**27.03 / 0.7646**

Table 5. Quantitative comparison of **BD** and **DN** degradation models. Best results are **highlighted**.

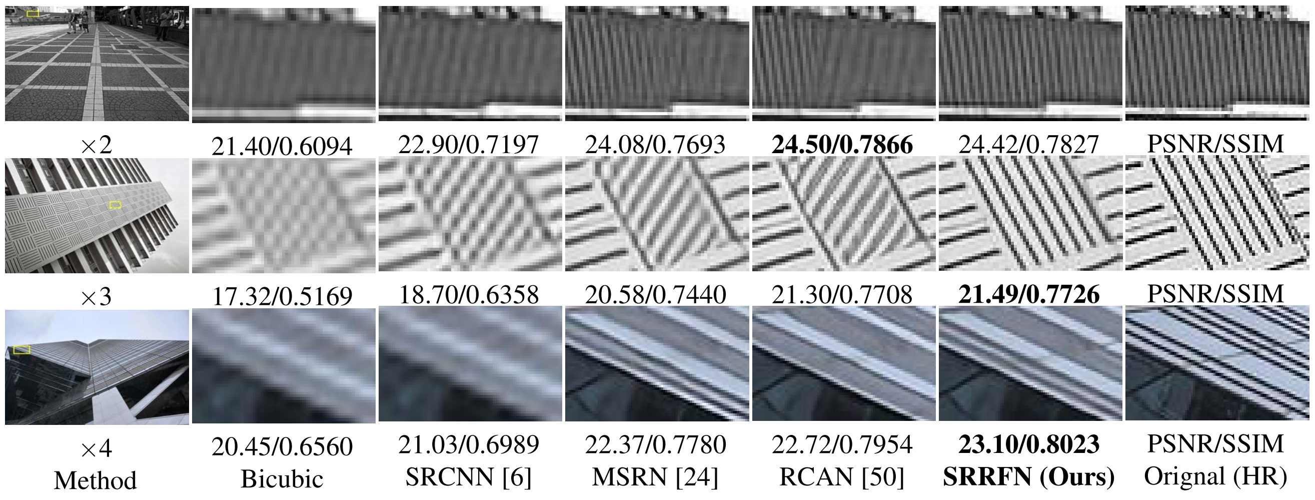

Figure 4. Visual comparison for ×2, ×3, and ×4 SR images. Our SRRFN can reconstruct realistic images with sharp edges.

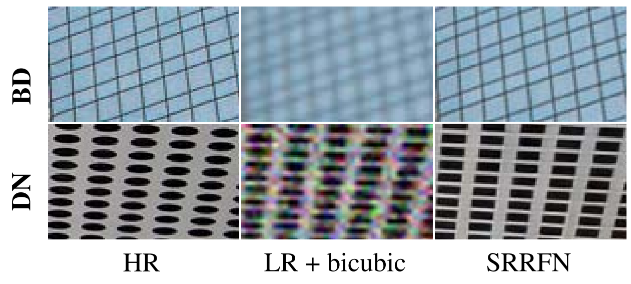

Figure 5. Results on **BD** and **DN** degradation models with x3.

RDN [51], SRFBN [25], and RCAN [50]. All of these models have been retrained by relevant degradation LR-HR images. Obviously, our SRRFN achieves state-of-the-art results on all test datasets. This demonstrates that our SRRFN can better deal with complex downsampled LR images. It also means that the fractal structure has better fault tolerance and robustness. In Figure 5, we show the effect of our SRRFN in **BD** and **DN** models. Obviously, SRRFN achieves superior performance even if LR images are degraded more seriously. All reconstructed SR images can be downloaded from `https://dwz.cn/noGR4Flb`.

5. Investigations

5.1. Differences with Previous Work

Previous works focused on the design of the network structure. Therefore, plenty of well-designed networks have been proposed. However, we aim to explore a simple SR framework that reuses the well-tested network modules and simplifies the design process. SRRFN and RCAN [50] seem to have similar structures, but they have at least the following differences: (1). our proposed fractal module (FM) has no fixed structure, we use the residual block as the basic component just to give a simple example; (2). the RCAN is difficult to expand, and the introduced CAM destroys the fractal structure; (3). the SRRFN introduces the recursive mechanism for recursive residual learning to maximize the use of model parameters and highly complex feature learning; (4). the parameters of SRRFN are only 1/4 of RCAN and the execution time is 3 times faster than it.

5.2. Study of Fractal Depth (D)

The fractal depth will greatly affect the parameter quantity and performance of the fractal module. In Figure 6.(A), we show the impact of D on the model performance. We set $K = 3$ and $N_1 = N_2 = \cdots = N_D = 4$. Obviously, the PSNR result increases as D increases. Meanwhile, this growth rate decreases as D increases. This verifies the view of "performance bottleneck" we presented in Sec.1.(1). It cannot be ignored that the parameter quantity will increase as D increases. Therefore, D can be selected according to actual demands. We set $D = 3$ in this paper to achieve a good balance between the model size and performance.

5.3. Study of Recursive Stage (S)

We introduce recursive mechanism into SRRFN to further improve the model performance. In Figure 6.(B), we show the effect of S on the model performance and execution time. We can observe that as S increases, the PSNR of SRRFN first increases and finally stabilizes. Meanwhile, as S increases, the execution time will gradually increase. Therefore, we conclude that although the recursive mechanism is effective, the number of S should be considered. In this paper, we set $S = 4$ to achieve the best balance between the model performance and execution time.

5.4. Model Size and Execution Time

Various large size SR models have been proposed in recent years. These models are getting deeper and wider, with a large number of parameters or low execution speed. We show the comparison of model parameters and execution time between SRRFN and other SR models in Figure 6.(C) and (D), respectively. We can clearly see that the parameters of SRRFN are 1/10 of EDSR [26], 1/5 of RDN [51], and 1/4 of RCAN [50]. Among these state-of-the-art SR models, our SRRFN achieves superior results with fewer parameters. Regarding the execution time, SRRFN also shows excellent results. The execution time of SRRFN is close to those lightweight SR models (e.g. SRCNN [6], VDSR [18], MSRN [24]), but the performance is far superior to them. It is worth noting that SRRFN is three times faster than RCAN [50], but the performance is almost the same. Moreover, as the size of the LR image increases, the time gap will gradually increase since the CAM introduced in RCAN [50] consumes a lot of time. All these investigations show that SRRFN can achieve excellent performance with fewer parameters and faster execution time.

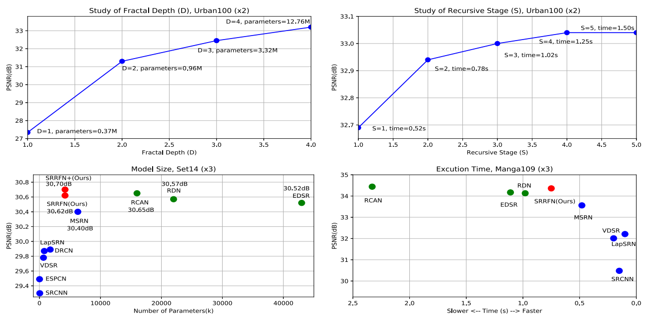

Figure 6. Investigations of fractal depth (D), recursive stage (S), model size, and execution time in **BI** model.

6. Discussion

Benefits of SRRFN: We combine the fractal structure with CNN to construct the fractal module, which greatly simplifies the model design and can construct an infinite variety of topological structures through a simple basic component. These topologies structure provide a large number of search paths that enable the network to extract abundant image features to reconstruct high-quality SR images.

Limitations of SRRFN: SRRFN consists of a basic component and a hyperparameter (D), which determines its final structure. However, which module to choose as the basic component and how to set the fractal depth are worth exploring. In this paper, we use a simple module as the basic component and set $D = 3$ in the final SRRFN. Although this solution has achieved excellent results, there may be better solutions. In future works, we aim to introduce the AutoML [27, 52] into the FM to automatically select and determine the final basic component and fractal depth.

7. Conclusions

In this paper, we proposed a Super-Resolution Recursive Fractal Network (SRRFN). This is a lightweight and accurate SR framework. SRRFN introduces the fractal module (FM) for feature extraction and uses recursive mechanism for recursive residual learning, which achieves competitive results with fewer parameters and faster execution time. The FM is a flexible module with self-similarity and infinitely refined structure, which can be defined by a simple basic component and generated by recursion and iteration. This special structure makes our model more fault-tolerant and robust. Furthermore, this module can also be used in other low-level computer vision tasks (e.g. image denoising and image dehazing) for feature extraction. We will further verify the performance of the FM in future works.

Acknowledgments. This work is sponsored by the key project of the National Natural Science Foundation of China (No. 61731009), the National Science Foundation of China (No. 61501188), and the "Chenguang Program" supported by Shanghai Education Development Foundation.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, pages 1110–1121, 2017.

[2] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[3] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *The British Machine Vision Conference*, 2012.

[4] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.

[6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2014.

[7] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 391–407, 2016.

[8] Chen Du, He Zewei, Sun Anshun, Yang Jiangxin, Cao Yanlong, Cao Yanpeng, Tang Siliang, and Michael Ying Yang. Orientation-aware deep neural network for real image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 0–0, 2019.

[9] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *arXiv:1904.01169*, 2019.

[10] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1664–1673, 2018.

[11] Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. *arXiv:1903.10128*, 2019.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017.

[15] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015.

[16] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015.

[17] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4565–4574, 2016.

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016.

[19] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645, 2016.

[20] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5835–5843, 2017.

[21] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 624–632, 2017.

[22] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[23] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4681–4690, 2017.

[24] Juncheng Li, Faming Fang, Kangfu Mei, and Guixu Zhang. Multi-scale residual network for image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[25] Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu. Feedback network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[26] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single

image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, pages 1132–1140, 2017.

[27] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[28] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[29] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv:1412.6632*, 2014.

[30] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 416–423, 2001.

[31] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017.

[32] Kangfu Mei, Aiwen Jiang, Juncheng Li, Jihua Ye, and Mingwen Wang. An effective single-image super-resolution model using squeeze-and-excitation networks. In *International Conference on Neural Information Processing*, pages 542–553, 2018.

[33] Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. Dual attention networks for multimodal reasoning and matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 299–307, 2017.

[34] 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. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[35] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 5, 2017.

[36] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of thee IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[37] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4539–4547, 2017.

[38] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Asian Conference on Computer Vision*, pages 111–126, 2014.

[39] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[40] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[41] Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 370–378, 2015.

[42] Jae Woong Soh, Gu Yong Park, Junho Jo, and Nam Ik Cho. Natural and realistic single image super-resolution with explicit natural manifold discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[43] Xiangyu Xu, Yongrui Ma, and Wenxiu Sun. Towards real scene super-resolution with raw images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[44] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv:1511.07122*, 2015.

[45] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, pages 711–730, 2010.

[46] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3929–3938, 2017.

[47] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 6, 2018.

[48] Xuaner Zhang, Qifeng Chen, Ren Ng, and Vladlen Koltun. Zoom to learn, learn to zoom. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[49] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[50] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[51] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[52] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv:1611.01578*, 2016.