FC^{2}N: Fully Channel-Concatenated Network for Single Image Super-Resolution

Xiaole Zhao[1], Ying Liao[1], Ye Li[1], Tao Zhang[1,2,3], Xueming Zou[1,2,3]
[1] School of Life Science and Technology, University of Electronic Science and Technology of China
[2] High Field Magnetic Resonance Brain Imaging Laboratory of Sichuan, Chengdu, China
[3] Key Laboratory for NeuroInformation of Ministry of Education, Chengdu, China
zxlation@foxmail.com, taozhangjin@gmail.com, mark.zou@alltechmed.com

Abstract

Most current image super-resolution (SR) methods based on deep convolutional neural networks (CNNs) use residual learning in network structural design, which contributes to effective back propagation, thus improving SR performance by increasing model scale. However, deep residual network suffers some redundancy in model representational capacity by introducing short paths, thus hindering the full mining of model capacity. In addition, blindly enlarging the model scale will cause more problems in model training, even with residual learning. In this work, a novel network architecture is introduced to fully exploit the representational capacity of the model, where all skip connections are implemented by weighted channel concatenation, followed by a 1×1 conv layer. Based on this weighted skip connection, we construct the building modules of our model, and improve the global feature fusion (GFF). Unlike most previous models, all skip connections in our network are channel-concatenated and no residual connection is adopted. It is therefore termed as fully channel-concatenated network (FC^{2}N). Due to the full exploitation of model capacity, the proposed FC^{2}N achieves better performance than other advanced models with fewer model parameters. Extensive experiments demonstrate the superiority of our method to other methods, in terms of both quantitative metrics and visual quality.

1. Introduction

Single image super-resolution (SR) is a classic problem in low-level computer vision that aims at reconstructing a high-resolution (HR) image from one single low-resolution (LR) image. Although a lot of solutions have been proposed for image SR, it is still an active yet challenging research topic in computer vision community due to its ill-poseness nature and high practical values [3, 45, 50].

In recent years, deep learning techniques [23], especially convolutional neural networks (CNNs) [24, 25] and residual learning [11], have greatly promoted the best performance

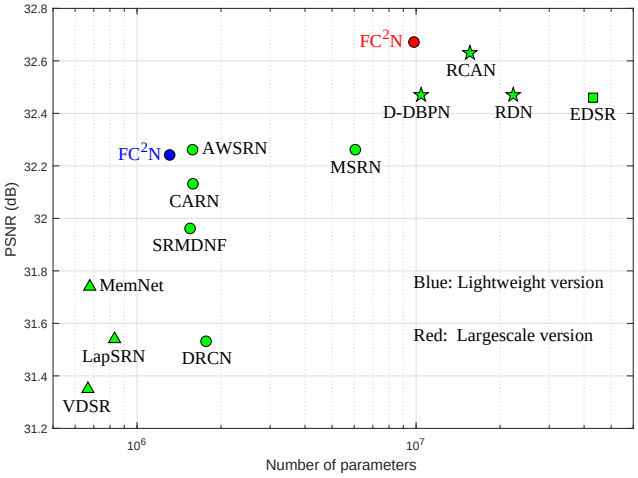


Figure 1. Performance comparison between several typical SISR networks on Set5 [4] with SR×4. The symbols △, ○, ☆ and □ represent the models with less than 1M, 10M, 20M, and more than 20M parameters respectively.

of image SR. A representative work that successfully adopts CNNs to SR problem is SRCNN [7], which is a three-layer network that can learn an end-to-end mapping between LR and HR images and achieve satisfactory performance at that time. Subsequently, many studies were conducted to design and build more accurate and efficient SR networks, such as [20], [35], [19], [50], [29], [48], [46] etc. One of the major trends in these models is that networks get deeper and more complex for further SR performance gain. With the increase of network scale, the training difficulty caused by gradient vanishing/explosion becomes more serious, and more tricks are needed to ensure effective model training [27]. Residual learning [11] is probably one of the most commonly-used techniques to ease this training difficulty, which is a simple element-wise addition of features at different layers. It can help extend the model to previously unreachable depths and capabilities by introducing short paths that carries gradients throughout the extent of very deep models [38]. These short paths, however, result in a large amount of representational

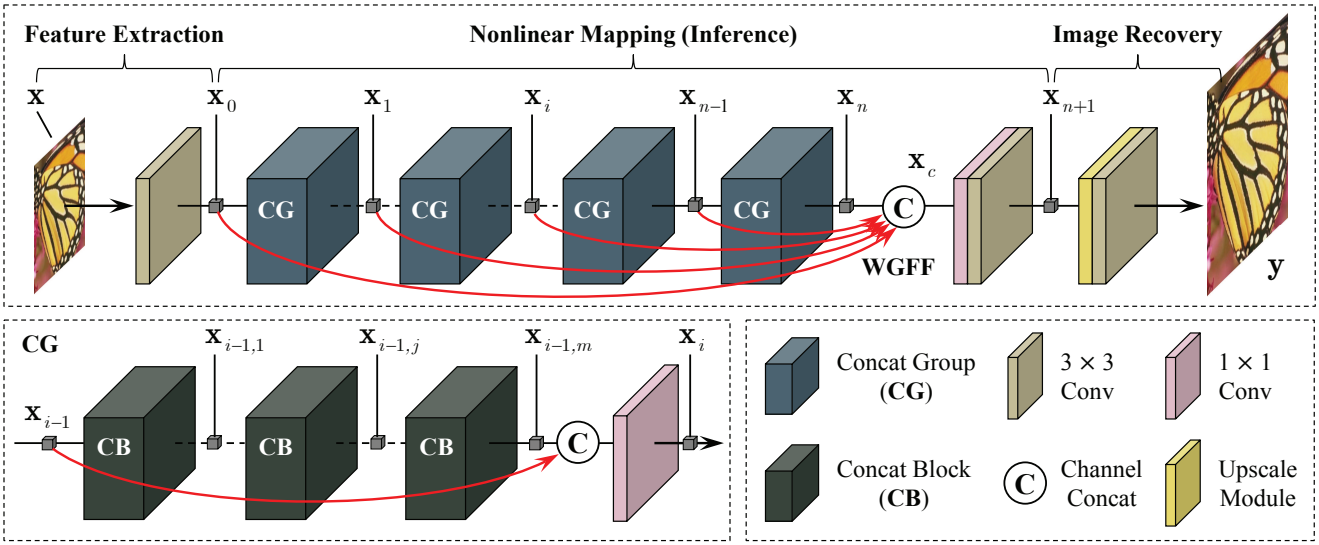


Figure 2. Overall network architecture of the proposed FC^{2}N model. The red arrows in weighted global feature fusion (WGFF) and concat group (CG) represent weighted channel concatenation. Note that there are no residual connections in the entire network.

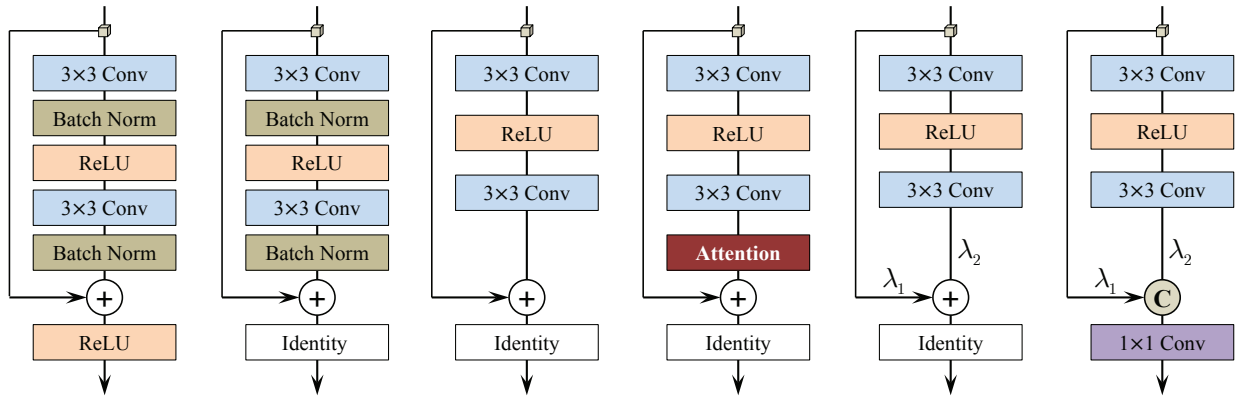
redundancy in deep residual networks [16], thus hindering the full mining of model capabilities.

To fully utilize hierarchical features and further exploit model capacities, many SR models tend to combine residual learning with channel-wise concatenation, which is widely believed to help new feature exploration and learning good representations [15, 5, 14, 50], e.g., MemNet [36], AWSRN [39], RDN [48], MSRN [27], CARN [2], CSSFN [49] and DBDN [40] etc. On the one hand, most of these SR models use residual learning [11, 12] for stable and effective model training, but as mentioned above, it is not conducive to full exploration of model capacities. On the other hand, channel concatenation in these models is adopted to directly connect different layers, which ignores the contribution of adaptive connection strength to model capacities, namely, weighted channel concatenation. Although the weighted connections are considered to impede effective back propagation in case of residual learning [12], we believe that they are more in line with the way neurons behave in the human brain, thus more physiologically sound.

Considering the above problems, we introduce a novel network for single image SR tasks in this work, in which all skip connections are implemented by channel concatenation without any residual connection. In addition, each branch of these channel concatenations are attached by a weighting factor to further explore the representational capacity of the model. Based on such weighted channel concatenation, we construct the building modules of the network, including concat block (CB) and concat group (CG) that are used for effective local feature utilization. Moreover, we introduce the weighted version of global feature fusion (GFF) [48] by weighting the branches of GFF's skip connections, i.e., WGFF. All these building modules are built based on our weighted channel concatenation. Therefore, the network is termed as fully channel-concatenated network (FC^{2}N). The main contributions of this work are as following:

- We present a novel CNN architecture for single image SR, in which channel concatenation is used to conduct interlayer bypasses. Due to the full mining of model representation ability, it has good performance in both lightweight and large-scale implementations.
- A new building unit constructed by weighted channel concatenation is introduced, i.e., CB. We demonstrate that the residual block [29] that is widely adopted in SR models is a special case of the CB block.
- Based on the global feature fusion (GFF) proposed by Zhang *et al.* [48], a weighted implementation of GFF is explored, which makes the utilization of hierarchical features more flexible and reasonable, thus promoting the representational capacity of the model.
- We further construct a concat group (CG) module for local information fusion, which consists of a group of cascaded CBs. The weighted channel concatenation is also used to connect the input of the first CB and output of the last CB in a CG, followed by a 1×1 conv layer for local information integration.

The remainder of this paper is organized as follows. In section 2, we introduce some previous works related to this paper. The illustration of the proposed FC^{2}N is presented in section 3. Next, the experimental results and analyses are given in section 4. Finally, we conclude the whole paper in section 5.



(a) ResNet [11, 12] (b) SRResNet [26] (c) EDSR [29] (d) Attention [46, 14] (e) AWSRN [39] (f) Concat Block

Figure 3. Several typical building blocks in deep CNN models. "**C**" represents channel-wise concatenation, and "**+**" denotes element-wise addition. "Attention" denotes spatial or channel attention mechanism, and "Identity" means identity mapping.

2. Related Work

2.1. Image super-resolution with deep learning

The pioneering work that uses deep learning techniques to solve single image super-resolution tasks in the modern sense is SRCNN [7], which is a three-layer CNN that maps LR images to HR images in an end-to-end manner. Through introducing global residual (GRL) learning, Kim *et al.* [19] increased the network depth up to 20 layers and achieved significant improvement in SR performance. Tai *et al.* [36] presented a very deep memory network (MemNet) to solve the problem of long-term dependency. Instead, some other works, e.g, DRCN [20] and DRRN [35], focused on weight sharing to reduce the scale of model parameters. Although these methods achieve superior performance, they require the bicubic-interpolated version of the original LR images as input, which inevitably loses some details and increases computational burden greatly [33, 46].

This problem can be alleviated by placing nonlinear inference in LR image space and upscaling image resolution at the end of the network, such as transpose convolution [8] and ESPCNN [33]. Benefitting from this, some models can improve SR performance by significantly increasing model scale, e.g., EDSR [29], RDN [48], D-DBPN [10] and RCAN [46] etc. However, the performance gain of these methods depends largely on the increase of model scale, e.g., EDSR [29] has about 43M model parameters and 70-layer network depth, and RCAN [46] also has more than 16M parameters and 400-layer network depth.

To generate more realistic SR results, especially for large SR scaling factors, Ledig *et al.* [26] proposed to introduce generative adversarial network (GAN) [9] into single image SR framework (SRGAN). They developed a new network structure based on ResNet [11, 12] (SRResNet) and treated it as the generator of a GAN [9] with perceptual loss [18]. The idea was also introduced in EnhanceNet [32] that combined automated texture synthesis and perceptual loss. Although these GAN-based models can ease over-smoothing artifacts to some degree, their predicted results may not be faithfully reconstructed with unpleasing artifacts [46].

2.2. Interlayer bypass connections

A simple and direct manner to improve the performance of deep models is to increase model scale, such as network depth and width. However, more problems will arise as the scale of deep models increases, and more training tricks are needed to ensure effective model training [27]. To ease the training difficulty caused by the increased model scale, skip connections are widely used in network design. Two typical skip connections are residual connection [11, 12] and channel concatenation [15]. However, although residual connection is widely applied in SR models, such as [19], [35], [36], [29] and [48], [46], there is a large amount of representational redundancy in residual networks [16, 37], which indicates that residual learning may hinder the full mining of model capacities. In fact, when the scale of the models is relatively fixed, the performance of residual networks still has potential to be further improved [2, 39].

Channel concatenation is another way to implement skip connections in the context of image SR, such as MemNet [36], SRDenseNet [37], RDN [48], MSRN [27], as well as AWSRN [39] etc. However, these methods usually combine channel concatenation with residual connections, expecting to fully use intermediate features and mitigate the training difficulty. Instead, in the proposed FC^{2}N network, all skip connections are implemented by channel concatenation in a weighted manner. Therefore, the network can adaptively construct a reasonable amount of skip connections with appropriate strength, thereby fully mining the representational capacity of the model.

3. Fully Channel-Concatenated Network

3.1. Weighted channel concatenation

Most current deep models are modularized architectures that consist of many stacked building blocks, e.g., ResNet [11], MemNet [36], DRRN [35], SRResNet [26], EDSR [29], AWSRN [39], RCAN [46] etc. The structure of some typical building blocks is outlined in Fig.3. In the context of image SR, Conv-ReLU-Conv based *residual block* [29] and its variants are broadly adopted as the building modules of deep SR models, such as Fig.3(d) and Fig.3(e). Most these building blocks, however, are combined with the strategy of residual learning for efficient model training.

The building block of the proposed FC^{2}N model is also based on the Conv-ReLU-Conv structure, but it avoids using residual learning. Instead, we adopt channel concatenation [15] followed by a 1×1 conv to fuse the input and output of the nonlinear mapping branch, in a weighted way as shown in Fig.3(f). Temporarily, let $\mathbf{x}_t \in \mathbb{R}^{H\times W\times C}$ be the input of a CB and $\mathcal{H}(\cdot)$ the function corresponding to the nonlinear mapping branch, the weighted channel-wise concatenation can be formulated as:

$$\mathbf{x}_{t+1} = \mathcal{L}([\lambda_1\mathbf{x}_t, \lambda_2\mathcal{H}(\mathbf{x}_t)]), \tag{1}$$

where $\mathbf{x}_{t+1}$ is the output of the CB, and $[\dots]$ represents the operation of channel-wise concatenation. $\mathcal{L}(\cdot)$ corresponds to the 1×1 conv layer, and λ_1 and λ_2 are the corresponding weighting factors, as shown in Fig.3(f).

Let's rewrite $\mathbf{x}_t = [x_t^1, \dots, x_t^i, \dots, x_t^C]$ and $\mathcal{H}(\mathbf{x}_t) = [x_h^1, \dots, x_h^j, \dots, x_h^C]$, both of which have C feature maps with spatial size of $H \times W$. Denote the kernel of the 1×1 conv as $\mathbf{K} \in \mathbb{R}^{2C\times C}$ with $2C$ input channels and C output channels[1]. Omitting the biases, then $\mathbf{x}_{t+1}$ is obtained by:

$$x_{t+1}^u = \sum_{i=1}^{C} \lambda_1 x_t^i \mathbf{K}(i,u) + \sum_{j=1}^{C} \lambda_2 x_h^j \mathbf{K}(C+j,u), \tag{2}$$

where x_{t+1}^u is the u-th feature map of $\mathbf{x}_{t+1}$, i.e., $\mathbf{x}_{t+1} = [x_{t+1}^1, \dots, x_{t+1}^u, \dots, x_{t+1}^C]$. Here u is the index of output channel. Now we give that both Fig.3(c) and Fig.3(e) are special cases of our CB blocks, i.e., Fig.3(f). If $\mathbf{K}$, λ_1 and λ_2 satisfy: $\mathbf{K}(i,i) = \mathbf{K}(C+i,i) = 1, i = 1,\dots,C$, all other elements in $\mathbf{K}$ are 0, and $\lambda_1 = \lambda_2 = 1$, then a CB block degrades to the residual block of EDSR [29]. If λ_1 and λ_2 act as learnable weighting factors at this time, then it degrades to the residual block in AWSRN [39].

Moreover, a CB block can also achieve channel attention mechanism [13, 46], which can be viewed as a guidance to bias the allocation of available processing resources towards the most informative components of an input [13]. Unlike previous self-attention, the 1×1 conv layer contributes to the simultaneous attention to the input and output features of the nonlinear mapping branch.

[1] In implementation, $\mathbf{K}$ has the shape of $[k, k, C_{\text{in}}, C_{\text{out}}]$, where k is kernel size, C_{in} and C_{out} denote the number of input channels and output channels, respectively. Since the spatial size of $\mathbf{K}$ is 1×1 here, we remove the singular dimensions for simplification.

3.2. Overall network structure

The overall structure of the FC^{2}N network is shown in Fig.2. Similar to many previous models, it mainly includes 3 stages: shallow feature extraction, nonlinear mapping and image reconstruction, which are denoted as $\mathcal{F}_E(\cdot)$, $\mathcal{F}_N(\cdot)$ and $\mathcal{F}_R(\cdot)$ respectively. Assume that the model takes $\mathbf{x}$ as input and outputs $\mathbf{y}$. The shallow features are first extracted by a single 3×3 conv layer:

$$\mathbf{x}_0 = \mathcal{F}_E(\mathbf{x}), \tag{3}$$

where $\mathbf{x}_0$ is the extracted shallow feature maps, and $\mathcal{F}_E(\cdot)$ denotes the 3×3 conv layer. Subsequently, $\mathbf{x}_0$ is input into the nonlinear mapping subnet for inference, generating deep feature maps. This can be formulated as:

$$\mathbf{x}_{n+1} = \mathcal{F}_N(\mathbf{x}_0), \tag{4}$$

where $\mathbf{x}_{n+1}$ is the generated deep feature maps, and $\mathcal{F}_N(\cdot)$ corresponds to the entire nonlinear mapping stage, which consists of n cascaded concat groups (CG) combined with a weighted global feature fusion (WGFF). Let's denote the i-th CG as $\mathcal{G}_i(\cdot)$ $(i = 1, 2, \dots, n)$. As shown in Fig.2, we can get n intermediate features as following:

$$\mathbf{x}_i = \mathcal{G}_i(\mathbf{x}_{i-1}), \quad i = 1, 2, \dots, n. \tag{5}$$

Therefore, we can obtain the nonlinear mapping from $\mathbf{x}_0$ to $\mathbf{x}_n$ iteratively:

$$\mathbf{x}_n = \mathcal{G}_n(\mathbf{x}_{n-1}) = \mathcal{G}_n(\mathcal{G}_{n-1}(\cdots(\mathcal{G}_1(\mathbf{x}_0))\cdots)). \tag{6}$$

These intermediate features are concatenated together along the channel direction, and then compressed by a 1×1 conv layer. Finally, a 3×3 conv layer is used to generate the final deep features $\mathbf{x}_{n+1}$:

$$\mathbf{x}_{n+1} = \mathcal{F}_D(\mathbf{x}_c) = \mathcal{F}_D([\lambda_0\mathbf{x}_0, \lambda_1\mathbf{x}_1, \dots, \lambda_n\mathbf{x}_n]), \tag{7}$$

where λ_i denotes the weighting factor of the corresponding intermediate feature. $[\dots]$ represents channel concatenation operation, and $\mathcal{F}_D(\cdot)$ denotes the function corresponding to the 1×1 conv layer followed by a 3×3 conv layer. Unlike [48] that directly fuses intermediate features, the weighted version of GFF can further explore model representational capacity by adding negligible model parameters.

At this time, the whole feature extraction and nonlinear inference is completed in the LR image space. An image reconstruction subnet is then integrated into the end of the network. As in [29, 48, 50], we adopt the ESPCNN [33] to

upsample the deep feature maps, and a linear transformation to recovery the HR image $\mathbf{y}$:

$$\mathbf{y} = \mathcal{F}_R(\mathbf{x}_{n+1}), \tag{8}$$

where $\mathbf{y}$ is the final SR output, and $\mathcal{F}_R(\cdot)$ corresponds to the upscale module followed by a 3×3 conv layer.

3.3. Concat group

A concat group is simply composed of m stacked concat blocks (CB), as shown in Fig.2. Let $\mathbf{x}_{i-1} = \mathbf{x}_{i-1,0}$ be the input of the first CB in the i-th CG, then we can obtain the local features as following:

$$\mathbf{x}_{i-1,j} = \mathcal{B}_{i,j}(\mathbf{x}_{i-1,j-1}), \quad j = 1, 2, \ldots, m, \tag{9}$$

where $\mathcal{B}_{i,j}(\cdot)$ is the function corresponding to the j-th CB in the i-th CG. Iteratively, we have:

$$\mathbf{x}_{i-1,m} = \mathcal{B}_{i,m}\Big[\mathcal{B}_{i,m-1}\Big(\cdots \mathcal{B}_{i,1}(\mathbf{x}_{i-1,0})\cdots\Big)\Big]. \tag{10}$$

To facilitate the information flow of the network, we further adopt channel concatenation to merge the input of the first CB and the output of the last CB, as shown in Fig.2. Thus we get the final output of the i-th CG:

$$\mathbf{x}_i = \mathcal{L}_i([\lambda_{i-1,0}\mathbf{x}_{i-1,0},\ \lambda_{i-1,m}\mathbf{x}_{i-1,m}]), \tag{11}$$

where $\mathcal{L}_i(\cdot)$ denotes the function corresponding to the 1×1 conv layer, and $[\ldots]$ denotes the channel concatenation, as in (7). $\lambda_{i-1,0}$ and $\lambda_{i-1,m}$ are two weighting factors. These two operations are similar to local feature fusion (LFF) in some previous work, e.g., [48] and [46]. However, we use weighted channel concatenation followed by a 1×1 conv layer, instead of a 3×3 conv layer followed by a residual connection [46], to fuse local features.

3.4. Training objective

Determining the network parameters is achieved by minimizing the loss (objective) function between the recovered HR images $\mathcal{F}_{FC^2N}(\mathbf{x}; \mathbf{\Theta})$ and their corresponding ground truth images $\mathbf{y}$, where $\mathcal{F}_{FC^2N}(\cdot)$ denotes the entire FC^{2}N network, and $\mathbf{\Theta}$ is the parameter set of the model. Several typical loss functions have been studied in literatures, such as L_2 [7, 19, 36, 45], L_1 [29, 48, 46, 10, 22], and adversarial [26, 22, 10] losses etc. For demonstrating the effectiveness of our FC^{2}N model and the fairness of comparison, we choose to optimize L_1 loss as some previous works. Given a training set $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{|\mathcal{D}|}$, where $|\mathcal{D}|$ is the total number of training samples, the loss is given by:

$$L_1(\mathbf{\Theta}) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\left\|\mathcal{F}_{FC^2N}\left(\mathbf{x}^{(i)}; \mathbf{\Theta}\right) - \mathbf{y}^i\right\|_1, \tag{12}$$

which is optimized by Adam [21] algorithm. More details on model training will be shown in section 4.1.

3.5. Implementation details

We implement our FC^{2}N network by setting $m = 8$ and $n = 16$. For feature channels, we use *wide activation* [43] for the nonlinear mapping branch of a CB, i.e., it is set to be $\{32, 128, 32\}$. By shrinking the dimensions of the input and output features and extending the dimensions before ReLU layers, this configuration favors to activating more low-level information without increasing model parameters [43, 39]. Elsewhere, the number of feature maps is 32. Except for the 1×1 conv layers annotated in Fig.2 and Fig.3, all other conv layers are 3×3 conv, where zero-padding is applied to keep the size of features unchanged. For the upscale module, we follow the strategy of [29, 48, 46] and adopt ESPCNN [33] to upscale LR features to HR image space iteratively. The last conv layer of the entire network has 3 filters as it outputs RGB color images. For skip connections in GFF, CG and CB, all weighting factors are learnable and initialized as 1.0.

4. Experimental Results

4.1. Settings

As in [29, 48, 46, 10], we use 800 training images from DIV2K dataset [1] as our training set. Data augmentation is performed on the training images by randomly horizontal and vertical flips, as well as 90° rotations and data range complementarity. Five benchmark datasets, including Set5 [4], Set14 [44], B100 [30], Urban100 [17] and Manga109 [31], are used for model testing. The SR results are typically evaluated with PSNR and SSIM [41] on Y channel of transformed YCbCr space. For training, 48×48 LR image patches are extracted from LR images, while HR patch size corresponds to the scaling factors. Batch size is set to 16 as in previous works [29, 48, 46]. The objective is trained by the Adam optimizer [21] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The learning rate is initialized as 2×10^{-4} for all layers and halved for every 4×10^5 training steps. The FC^{2}N model is trained for 10^6 iterations in total.

4.2. Model analysis

4.2.1 Weighted channel concatenation

In this subsection, we first analyze the effects of weighted channel concatenation. To show the superiority of channel concatenation to residual learning, we set a baseline model for comparison, in which all skip connections in CGs and CBs are replaced by non-weighted residual connections and WGFF is changed to GFF [48]. This model corresponds to the "baseline [res]" in Fig.4. To illustrate the impact of the learnable weighting factors, we also investigate the ablation of whether the channel concatenations in CB, CG and GFF are weighted. Notations and quantitative results of different configurations are shown in Fig.4 and Table 1.

Table 1. Testing results of the ablation study on weighted channel concatenation. If channel concatenation is weighted by the learnable parameters, it is represented as "1", otherwise it is denoted as "0". All the models are tested on Set5 [4] for SR$\times$ 4 ($m = 16, n = 8$).

Configs	Concat Block	/	0	1	0	1	0	1	0	1
	Concat Group	/	0	0	1	1	0	0	1	1
	Weighted GFF	/	0	0	0	0	1	1	1	1
SR$\times$4	PSNR (dB)	32.53	32.62	32.63	32.59	32.64	32.62	32.62	32.65	32.67

Table 2. Impact of n and m on the performance. The results are calculated on Set5 [4] for SR$\times$ 4 (PSNR (dB) | params (M)).

$n\backslash m$	2	4	6	8
2	31.75|0.40	32.03|0.70	32.12|1.01	32.31|1.31
4	32.02|0.71	32.23|1.31	32.33|1.92	32.40|2.53
8	32.23|1.33	32.36|2.54	32.52|3.76	32.56|4.97
16	32.40|2.57	32.53|5.00	32.62|7.43	32.67|9.86

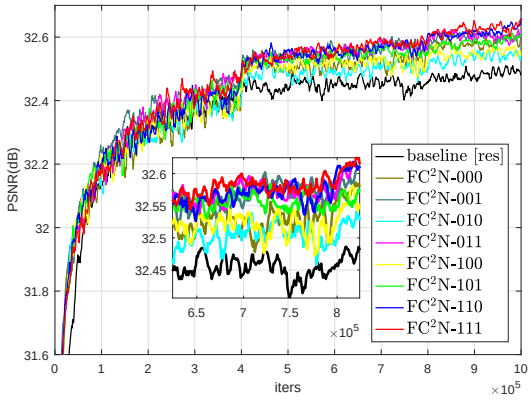


Figure 4. The comparison of validation SR performance between different configurations of weighted channel concatenation. These results are collected on Set5 [4] with SR$\times$4, corresponding to the testing results shown in Table 1.

Comparison to residual baseline As illustrated in Fig.4 and Table 1, by comparing with FC^{2}N-000, we can see that about 0.1dB performance gain can be achieved by changing the residual connections in CGs and CBs to non-weighted channel concatenations. The validation curves of baseline [res] and FC^{2}N-000 in Fig.4 also demonstrate that, after the learning rate decays, channel concatenation can improve the performance of the model more significantly than residual connection, although they have roughly the same number of model parameters (9.6M vs. 9.8M).

Learnable weighting factors In Table 1, by comparing FC^{2}N-001, FC^{2}N-010 and FC^{2}N-100 with FC^{2}N-000, we can observe that learnable weighting factors in both GFF and CBs favor to performance boosting, while in CGs they decrease the performance. However, when the learnability is activated in GFF (i.e., FC^{2}N-100 vs. FC^{2}N-110), model performance can be significantly improved by the weighting factors in CGs. At this time, weighting factors in CBs seem to have no significant contribution to model performance by comparing FC^{2}N-100 and FC^{2}N-101. In addition, the best performance given by FC^{2}N-111 exhibits that the weighted channel concatenation in GFF, CGs and CBs is beneficial to maximizing the performance gain. In particular, the WGFF regularly integrates shallow and deep features in an adaptive manner, thus improving the information flow throughout the entire network. This can also be verified by the comparison between FC^{2}N-011 and FC^{2}N-111.

The convergence curves shown in Fig.4 further verify the above analysis. In addition, it seems that weighting channel concatenation will slightly increase the instability of model training, but it indeed makes the model converge faster and achieve better performance.

4.2.2 Concat group and concat block

Table 2 exhibits the testing results of different combinations of m and n, on Set5 [4] with SR$\times$4. It can be observed that the increase in both m and n helps boost the performance of the model, which is unsurprising because increasing m and n obviously enlarges model scale, including network depth and model parameters. It is worth noting that at roughly the same model scale, larger m is more helpful to performance improvement than larger n. In addition, we can also observe from Table 2 that the proposed FC^{2}N network achieves excellent SR performance in both light-weight and large-scale implementations. This implies that it consistently provides good performance-scale tradeoffs as model scale changes.

4.3. Comparison with advanced methods

In this section, we compare the proposed FC^{2}N network with other advanced methods, such as SRCNN [7], DRCN [20], DRRN [35], VDSR [19], LapSRN [22], MemNet [36], EDSR [29], SRMDNF [45], D-DBPN [10], RDN [48] and RCAN [46] etc., quantitatively and qualitatively. Similar to [29], [48] and [46], we also use the geometric self-ensemble strategy to improve model performance and this is denoted as FC^{2}N$^+$. Furthermore, we introduce another strategy to further boost the performance of the model, termed as data range ensemble. During the testing time, we can generate an image $\bar{\mathbf{x}}$ with complementary data ranges for each testing image $\mathbf{x}$: $\bar{\mathbf{x}} = 255 - \mathbf{x}$, which gives us a corresponding HR output $\bar{\mathbf{y}}$ from the model. The final HR image of $\mathbf{x}$ is given by $[\mathbf{y} + (255 - \bar{\mathbf{y}})]/2$. When both strategies are applied, it is denoted as FC^{2}N^{++}.

Table 3. Quantitative comparison with other typical SR methods. The best and second best results in each comparative cell are marked in red and blue, respectively. Model parameters are also listed for comparison. r indicates SR scaling factor (PSNR (dB) / SSIM).

Method / Model	r	Param	Set5	Set14	B100	Urban100	Manga109
Bicubic	2	/	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	26.88 / 0.8403	30.80 / 0.9339
SRCNN [7]	2	57.3K	36.66 / 0.9542	32.45 / 0.9067	31.36 / 0.8879	29.50 / 0.8946	35.60 / 0.9663
DRCN [20]	2	1.77M	37.63 / 0.9588	33.04 / 0.9118	31.85 / 0.8942	30.75 / 0.9133	37.63 / 0.9723
DRRN [35]	2	0.30M	37.74 / 0.9591	33.23 / 0.9136	32.05 / 0.8973	31.23 / 0.9188	37.92 / 0.9760
VDSR [19]	2	0.67M	37.53 / 0.9590	33.05 / 0.9130	31.90 / 0.8960	30.77 / 0.9140	37.22 / 0.9750
LapSRN [22]	2	0.81M	37.52 / 0.9590	33.08 / 0.9130	31.80 / 0.8950	30.41 / 0.9100	37.27 / 0.9740
MemNet [36]	2	0.68M	37.78 / 0.9597	33.28 / 0.9142	32.08 / 0.8978	31.31 / 0.9195	37.72 / 0.9740
EDSR [29]	2	40.7M	38.11 / 0.9602	33.92 / 0.9195	32.32 / 0.9013	32.93 / 0.9351	39.10 / 0.9773
SRMDNF [45]	2	1.51M	37.79 / 0.9601	33.32 / 0.9159	32.05 / 0.8985	31.33 / 0.9204	38.07 / 0.9761
D-DBPN [10]	2	5.82M	38.09 / 0.9600	33.85 / 0.9190	32.27 / 0.9000	32.55 / 0.9324	38.89 / 0.9775
RDN [48]	2	22.1M	38.24 / 0.9614	34.01 / 0.9212	32.34 / 0.9017	32.89 / 0.9353	39.18 / 0.9780
RCAN [46]	2	15.4M	38.27 / 0.9614	34.11 / 0.9216	32.41 / 0.9026	33.34 / 0.9384	39.44 / 0.9786
FC^{2}N [Ours]	2	9.82M	38.29 / 0.9616	34.14 / 0.9224	32.40 / 0.9025	33.18 / 0.9379	39.51 / 0.9787
FC^{2}N$^+$ [Ours]	2	9.82M	38.33 / 0.9617	34.24 / 0.9224	32.44 / 0.9029	33.34 / 0.9388	39.62 / 0.9790
FC^{2}N^{++} [Ours]	2	9.82M	38.34 / 0.9618	34.25 / 0.9225	32.45 / 0.9030	33.39 / 0.9392	39.68 / 0.9792
Bicubic	3	/	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	24.46 / 0.7349	26.95 / 0.8556
SRCNN [7]	3	57.3K	32.75 / 0.9090	29.30 / 0.8215	28.41 / 0.7863	26.24 / 0.7989	30.48 / 0.9117
DRCN [20]	3	1.77M	33.82 / 0.9226	29.76 / 0.8311	28.80 / 0.7963	27.15 / 0.8276	32.31 / 0.9328
DRRN [35]	3	0.30M	34.03 / 0.9244	29.96 / 0.8349	28.95 / 0.8004	27.53 / 0.8378	32.74 / 0.9390
VDSR [19]	3	0.67M	33.66 / 0.9213	29.77 / 0.8314	28.82 / 0.7976	27.14 / 0.8279	32.01 / 0.9310
LapSRN [22]	3	0.81M	33.82 / 0.9227	29.87 / 0.8320	28.82 / 0.7980	27.07 / 0.8280	32.21 / 0.9350
MemNet [36]	3	0.68M	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	27.56 / 0.8376	32.51 / 0.9369
EDSR [29]	3	43.7M	34.65 / 0.9280	30.52 / 0.8462	29.25 / 0.8093	28.80 / 0.8653	34.17 / 0.9476
SRMDNF [45]	3	1.53M	34.12 / 0.9254	30.04 / 0.8382	28.97 / 0.8025	27.57 / 0.8398	33.00 / 0.9403
RDN [48]	3	22.3M	34.71 / 0.9296	30.57 / 0.8468	29.26 / 0.8093	28.80 / 0.8653	34.13 / 0.9484
RCAN [46]	3	15.6M	34.74 / 0.9299	30.65 / 0.8482	29.32 / 0.8111	29.09 / 0.8702	34.44 / 0.9499
FC^{2}N [Ours]	3	9.87M	34.76 / 0.9302	30.66 / 0.8485	29.31 / 0.8106	29.04 / 0.8700	34.63 / 0.9504
FC^{2}N$^+$ [Ours]	3	9.87M	34.85 / 0.9307	30.76 / 0.8495	29.36 / 0.8114	29.22 / 0.8725	34.87 / 0.9514
FC^{2}N^{++} [Ours]	3	9.87M	34.85 / 0.9307	30.78 / 0.8497	29.37 / 0.8115	29.24 / 0.8727	34.91 / 0.9515
Bicubic	4	/	28.42 / 0.8104	26.00 / 0.7027	25.96 / 0.6675	23.14 / 0.6577	24.89 / 0.7866
SRCNN [7]	4	57.3K	30.48 / 0.8628	27.50 / 0.7513	26.90 / 0.7101	24.52 / 0.7221	27.58 / 0.8555
VDSR [19]	4	0.67M	31.35 / 0.8830	28.02 / 0.7680	27.29 / 0.0726	25.18 / 0.7540	28.83 / 0.8870
LapSRN [22]	4	0.81M	31.54 / 0.8850	28.19 / 0.7720	27.32 / 0.7270	25.21 / 0.7560	29.09 / 0.8900
MemNet [36]	4	0.68M	31.74 / 0.8893	28.26 / 0.7723	27.40 / 0.7281	25.50 / 0.7630	29.42 / 0.8942
EDSR [29]	4	43.1M	32.46 / 0.8968	28.80 / 0.7876	27.71 / 0.7420	26.64 / 0.8033	31.02 / 0.9148
SRMDNF [45]	4	1.56M	31.96 / 0.8925	28.35 / 0.7787	27.49 / 0.7337	25.68 / 0.7731	30.09 / 0.9024
D-DBPN [10]	4	10.4M	32.47 / 0.8980	28.82 / 0.7860	27.72 / 0.7400	26.38 / 0.7946	30.91 / 0.9137
RDN [48]	4	22.3M	32.47 / 0.8990	28.81 / 0.7871	27.72 / 0.7419	26.61 / 0.8028	31.00 / 0.9151
RCAN [46]	4	15.6M	32.63 / 0.9002	28.87 / 0.7889	27.77 / 0.7436	26.82 / 0.8087	31.22 / 0.9173
FC^{2}N [Ours]	4	9.86M	32.67 / 0.9005	28.90 / 0.7889	27.77 / 0.7432	26.85 / 0.8089	31.41 / 0.9193
FC^{2}N$^+$ [Ours]	4	9.86M	32.73 / 0.9011	29.00 / 0.7906	27.83 / 0.7445	27.01 / 0.8127	31.71 / 0.9215
FC^{2}N^{++} [Ours]	4	9.86M	32.74 / 0.9012	29.02 / 0.7909	27.83 / 0.7446	27.03 / 0.8130	31.74 / 0.9216
Bicubic	8	/	24.40 / 0.6580	23.10 / 0.5660	23.67 / 0.5480	20.74 / 0.5160	21.47 / 0.6500
SRCNN [7]	8	57.3K	25.33 / 0.6900	23.76 / 0.5910	24.13 / 0.5660	21.29 / 0.5440	22.46 / 0.6950
VDSR [19]	8	0.67M	25.93 / 0.7240	24.26 / 0.6140	24.49 / 0.5830	21.70 / 0.5710	23.16 / 0.7250
LapSRN [22]	8	0.81M	26.15 / 0.7380	24.35 / 0.6200	24.54 / 0.5860	21.81 / 0.5810	23.39 / 0.7350
EDSR [29]	8	45.5M	26.96 / 0.7762	24.91 / 0.6420	24.81 / 0.5985	22.51 / 0.6221	24.69 / 0.7841
D-DBPN [10]	8	23.2M	27.21 / 0.7840	25.13 / 0.6480	24.88 / 0.6010	22.73 / 0.6312	25.14 / 0.7987
RCAN [46]	8	15.7M	27.31 / 0.7878	25.23 / 0.6511	24.98 / 0.6058	23.00 / 0.6452	25.24 / 0.8029
FC^{2}N [Ours]	8	9.90M	27.25 / 0.7833	25.10 / 0.6479	24.87 / 0.6016	22.72 / 0.6331	25.00 / 0.7937
FC^{2}N$^+$ [Ours]	8	9.90M	27.35 / 0.7880	25.27 / 0.6512	24.96 / 0.6041	22.94 / 0.6398	25.34 / 0.8005
FC^{2}N^{++} [Ours]	8	9.90M	27.35 / 0.7872	25.29 / 0.6517	25.01 / 0.6043	22.97 / 0.6407	25.38 / 0.8015

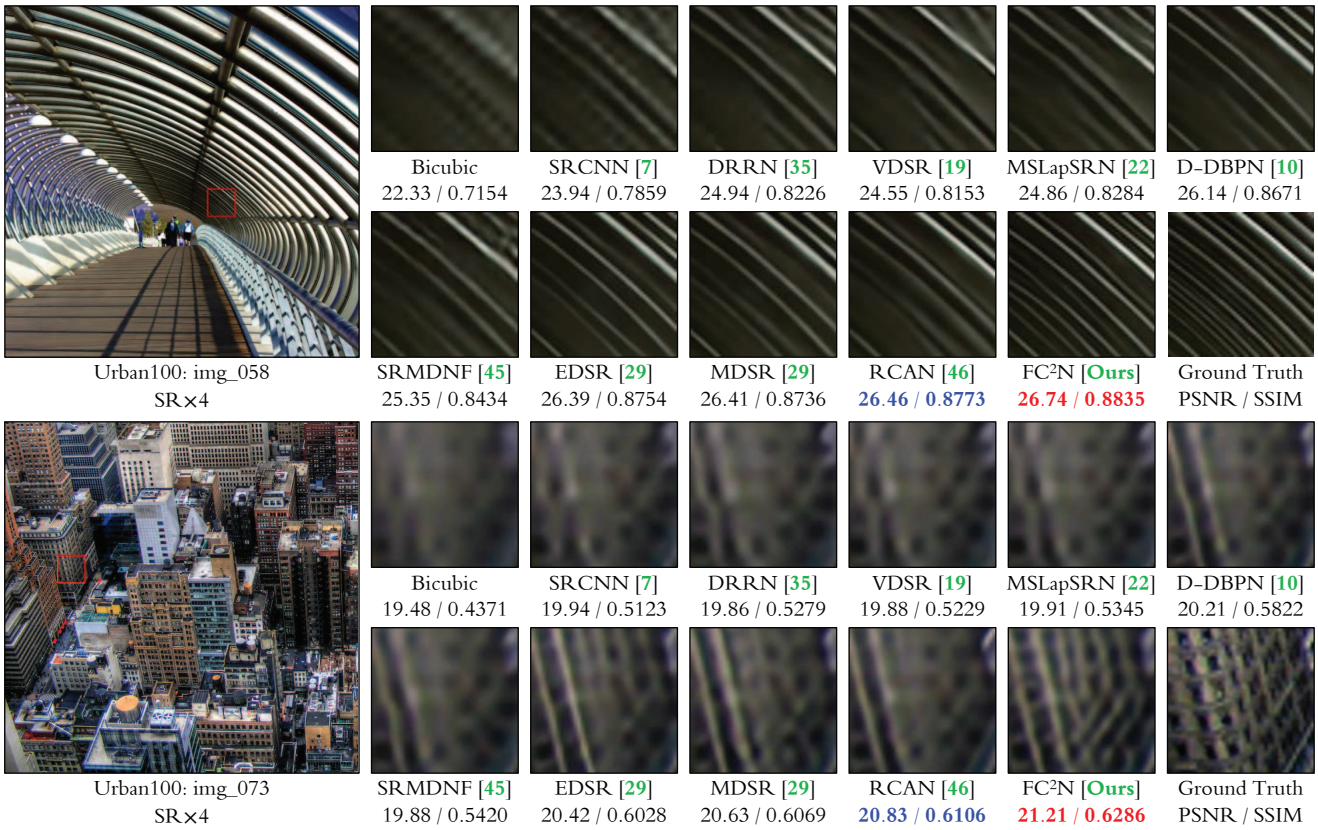


Figure 5. Visual comparison between several representational SR methods and our FC^{2}N model with $n = 16, m = 8$. The best and second best results are in red and blue, respectively.

4.3.1 Quantitative evaluation

Table 3 is the quantitative comparison between the proposed FC^{2}N model and other typical SR methods. It can be seen that the proposed FC^{2}N model outperforms most of other methods on all datasets. In particular, the FC^{2}N^{++} further improve the performance of the FC^{2}N^{+} on the whole, albeit slightly. This verifies the effectiveness of the incorporation of data range ensemble. When the scaling factor is 8, the performance of FC^{2}N becomes slightly worse than that of RCAN [46] but still better than many other SR models. The proposed FC^{2}N, however, has fewer model parameters than RCAN [46], which means that it provides a better trade-off between model performance and network scale.

4.3.2 Visual evaluation

Fig.5 shows the visual comparison between other typical SR methods and the FC^{2}N on 2 testing images from Urban100 [17] with SR×4. For "img_058", most of previous methods generate blurring artifacts at the fringes, especially for those in the lower left parts of the cropped images. However, only the FC^{2}N generate the result closer to the ground truth. For image "img_073", the blurring effect in the results of other methods in texture region is more obvious, but the FC^{2}N is still able to produce the result that can imply the potential structure more clearly. These comparisons illustrate the full exploitation of representational capacity of the model.

5. Conclusion

In this paper, we present a novel network structure aimed at effective image SR tasks, i.e., fully channel-concatenated network (FC^{2}N). Compared with most previous advanced SR methods, a major technical novelty of our FC^{2}N model is the introduction of weighted channel concatenation as all skip connections in the network, and the avoidance to utilize residual connections of element-level addition. Through the weighted channel concatenation, the network can not only adaptively select effective interlayer connections and make full use of hierarchical features, but also pay joint attention to the linear and nonlinear features, thus fully exploiting the representational capacity of the network. Extensive experiments show that our FC^{2}N model moves beyond most of the current state-of-the-art SR methods in both lightweight and large-scale implementations, which verifies its full mining of model representational capacity.

References

[1] E. Agustsson and R. Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPR*, pages 1122–1131, 2017. 5

[2] N. Ahn, B. Kang, and *et al*. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, pages 256–272, 2018. 2, 3, 11, 12, 14

[3] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Trans. Pattern Analy. Mach. Intell.*, 24(9):1167–1183, 2002. 1

[4] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, pages 1–10, 2012. 1, 5, 6, 11, 14, 16

[5] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. In *NIPS*, pages 4470–4478, 2017. 2

[6] X. Chu, B. Zhang, H. Ma, and *et al*. Fast, accurate and lightweight super-resolution with neural architecture search. *arXiv preprint*, arxiv:1901.06484, 2019. 11, 12

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016. 1, 3, 5, 6, 7, 11, 12

[8] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV 2016*, pages 391–407, 2016. 3, 11, 12

[9] I. J. Goodfellow, J. Pouget-Abadie, and *et al*. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 3

[10] M. Haris, G. Shakhnarovich, and N. Ukita. Deep back-projection networks for super-resolution. In *CVPR 2018*, pages 1664–1673, 2018. 3, 5, 6, 7

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 2, 3, 4, 14

[12] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645, 2016. 2, 3, 14, 15

[13] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks. *arXiv preprint*, arxiv:1709.01507, 2017. 4, 16

[14] Y. Hu, J. Li, Y. Huang, and X. Gao. Channel-wise and spatial feature modulation network for single image super-resolution. *arXiv preprint*, arXiv:1809.11130v1, 2018. 2, 3, 14, 16

[15] G. Huang, Z. Liu, and *et al*. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017. 2, 3, 4

[16] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661, 2016. 2, 3

[17] J. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015. 5, 8, 11

[18] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016. 3

[19] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR 2016*, pages 1646–1654, 2016. 1, 3, 5, 6, 7, 11, 12, 14

[20] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR 2016*, pages 1637–1645, 2016. 1, 3, 6, 7, 11, 12

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[22] W. S. Lai, J. B. Huang, N. Ahuja, and M. H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR 2017*, pages 5835–5843, 2017. 5, 6, 7, 11, 12

[23] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1

[24] Y. LeCun, B. E. Boser, and *et al*. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 1

[25] Y. LeCun, B. E. Boser, and *et al*. Handwritten digit recognition with a back-propagation network. In *NIPS*, pages 396–404, 1989. 1

[26] C. Ledig, L. Theis, F. Huszar, and *et al*. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR 2017*, pages 105–114, 2017. 3, 4, 5

[27] J. Li, F. Fang, K. Mei, and G. Zhang. Multi-scale residual network for image super-resolution. In *ECCV*, pages 527–542, 2018. 1, 2, 3

[28] J. Li, F. Fang, K. Mei, and G. Zhang. Multi-scale residual network for image super-resolution. In *ECCV 2018*, pages 527–542, 2018. 12

[29] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops 2017*, pages 1132–1140, 2017. 1, 2, 3, 4, 5, 6, 7, 11, 14, 15, 16

[30] D. R. Martin, C. C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, pages 416–425, 2001. 5, 11

[31] Y. Matsui, K. Ito, and *et al*. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017. 5, 11

[32] M. S. M. Sajjadi, B. Schölkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *ICCV*, pages 4501–4510, 2017. 3

[33] W. Shi, J. Caballero, F. Huszar, and *at al*. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR 2016*, pages 1874–1883, 2016. 3, 4, 5

[34] C. Szegedy, S. Ioffe, and *et al*. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint*, arxiv:1602.07261, 2016. 15

[35] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR 2017*, pages 2790–2798, 2017. 1, 3, 4, 6, 7, 11, 12

[36] Y. Tai, J. Yang, X. Liu, and C. Xu. MemNet: A persistent memory network for image restoration. In *ICCV 2017*, pages 4549–4557, 2017. 2, 3, 4, 5, 6, 7, 11, 12

[37] T. Tong, G. Li, X. Liu, and Q. Gao. Image super-resolution using dense skip connections. In *ICCV 2017*, pages 4809–4817, 2017. 3

[38] A. Veit, M. J. Wilber, and S. J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, pages 550–558, 2016. 1, 13, 14

[39] C. Wang, Z. Li, and J. Shi. Lightweight image super-resolution with adaptive weighted learning network. *arXiv preprint*, arxiv:1904.02358, 2019. 2, 3, 4, 5, 11, 12, 14, 15, 16

[40] Y. Wang, J. Shen, and J. Zhang. Deep bi-dense networks for image super-resolution. In *DICTA*, pages 1–8, 2018. 2

[41] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing*, 13(4):600–612, 2004. 5

[42] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *ICCV*, pages 370–378, 2015. 11

[43] J. Yu, Y. Fan, and *et al*. Wide activation for efficient and accurate image super-resolution. *arXiv preprint*, arxiv:1808.08718, 2018. 5, 14

[44] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Proc. 7th Int. Conf. Curves Surf.*, pages 711–730, 2010. 5, 11

[45] K. Zhang, W. Zuo, and L. Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR 2018*, pages 3262–3271, 2018. 1, 5, 6, 7, 11, 12

[46] Y. Zhang, K. Li, K. Li, and *et al*. Image super-resolution using very deep residual channel attention networks. In *ECCV 2018*, pages 294–310, 2018. 1, 3, 4, 5, 6, 7, 8, 14, 15, 16

[47] Y. Zhang, K. Li, B. Zhong, and *et al*. Residual non-local attention networks for image restoration. *arXiv preprint*, arxiv:1903.10082, 2018. 14

[48] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image super-resolution. In *CVPR 2018*, pages 2472–2481, 2018. 1, 2, 3, 4, 5, 6, 7, 11

[49] X. Zhao, H. Zhang, H. Liu, Y. Qin, T. Zhang, and X. Zou. Single MR image super-resolution via channel splitting and serial fusion network. *arXiv preprint*, arxiv:1901.06484, 2019. 2

[50] X. Zhao, Y. Zhang, T. Zhang, and X. Zou. Channel splitting network for single MR image super-resolution. *arXiv preprint*, arxiv:1810.06453, 2018. 1, 2, 4, 14, 16

Supplementary Material

The following items are contained in this supplementary material for further and more detailed illustration:

- Lightweight implementation of the FC^{2}N.
- Weighting factors in CGs and CBs.
- Illustration on network scale of the FC^{2}N.
- Analysis on the strategy of wide activation.
- Some discussions on related works.
- More visual comparisons with other SR methods.

A. Lightweight implementation

The weighted channel concatenation helps fully exploit the representational capacity of deep models, which implies that our FC^{2}N model should have good performance in both lightweight and large-scale implementations. To verify this point, we implement a lightweight FC^{2}N by simply setting $n = m = 4$, and compare it with several typical lightweight SR models, including SRCNN [7] and FSRCNN [8], VDSR [19], DRCN [20], LapSRN [22], DRRN [35], SRMDNF [45], SCN [42], MemNet [36], CARN [2], FALSR [6] and AWSRN [39] etc. Note here, we only compare lightweight models with almost the same scale of parameters, excluding those with large-scale model parameters, e.g., EDSR [29] and RDN [48] with 43M and 22M parameters, respectively.

In addition to model performance, an important factor to consider when applying SR models in real-world scenarios is model computation burden. Therefore, we also introduce MultAdds [2] as another quantitative evaluation index. For a conv layer, it is calculated as following:

$$\text{MultAdds} = k \times k \times C_{in} \times C_{out} \times H \times W, \qquad (13)$$

where k is the size of the conv kernel, and C_{in} and C_{out} are the input and output channels of this layer. H and W are the spatial size of output features. Note that in our lightweight FC^{2}N, except that n and m are set to 4, other configurations are the same as those for large-scale implementations.

Quantitative comparison The quantitative results of the compared lightweight models are shown in Table 4. These results are evaluated on four benchmark datasets, i.e., Set5 [4], Set14 [44], B100 [30] and Manga109 [31], and some of them are cited from [2] and [39]. As can be observed, our lightweight FC^{2}N model outperforms most of the compared methods in model performance, and achieves comparable performance to AWSRN [39]. However, our FC^{2}N network has fewer model parameters and computational operations than AWSRN [39]. Besides, the enhanced FC^{2}N model by the geometric and data range ensemble further improves the performance of the model, significantly surpassing AWSRN [39] with negligible extra computational effort.

Visual comparison Fig.6 displays the visual comparison between these lightweight models with SR$\times$4. As for the image "man" in Set14 [44], most these compared methods perform better than traditional bicubic interpolation, with sharper edges and more natural details. However, our FC^{2}N model presents the best visual effect, and the corresponding quantitative results also demonstrate its superiority to other methods. As for the image "img_008" in Urban100 [17], the advantage of the proposed FC^{2}N model is more obvious in that it presents the skylight grids with shape closest to the ground truth.

B. Learnable weighting factors

Adaptive weighted channel concatenation for interlayer bypass connections provides more flexibility to fully mine the representational capacity of the model, especially when the model scale is relatively fixed. In this section, we study the role of these learnable weighting factors.

Weighted global feature fusion For better insight of the weighted channel concatenation in GFF [48], we visualize the evolution curves and final values of the weights in Fig.7. Since the role of GFF is mainly embodied by the skip connections of CGs, instead of the shallow feature $\mathbf{x}_0$, we only focus on the weights of CG skip connections. The network studied here is with $n = 16, m = 8$ and SR$\times$4.

As shown in Fig.7(a), most of the weighting factors show a downward trend as model training proceeds, while a few factors increase first and then decrease, e.g., λ_{12}, λ_{13}, λ_{14} and λ_{15} that correspond to the relatively deep layers in the network. Fig.7(a) also demonstrates that the weights for the skip connections in the middle decrease faster than those at both ends, which may indicate that WGFF weights prefer to select shallow and deep features for effective global feature fusion. Fig.7(b) exhibits the final values of these weighting factors after the model training and similar results can also be observed. Intuitively, weighted channel concatenations in WGFF can help the model adaptively select and integrate intermediate features, thus improving the representational capacity and SR performance of the model.

Weights in concat group The evolution curves and final values of λ_1 and λ_2 for each CG (located at the end of each CG) are shown in Fig.8 and Fig.9. For λ_1 that weights the identity mapping, it shows an overall trend of increasing as the index of CGs increases. Besides, most of these weights have a relatively large determined value (> 0.5), as shown in Fig.8(b). For λ_2 that weights the nonlinear mapping (a series of cascaded CBs in this case), they display a similar

Table 4. Quantitative comparison between several lightweight models and our **lightweight** FC^{2}N ($n = m = 4$) on 4 benchmark datasets. The maximal value of each cell is marked in red and the second one is marked in blue (PSNR (dB) / SSIM).

Method	r	Param	MultAdds	Set5	Set14	B100	Manga109
Bicubic	2	/	/	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	30.80 / 0.9339
SRCNN [7]	2	57K	52.7G	36.66 / 0.9542	32.42 / 0.9063	31.36 / 0.8879	35.74 / 0.9661
FSRCNN [8]	2	12K	6.0G	37.00 / 0.9558	32.63 / 0.9088	31.53 / 0.8920	36.67 / 0.9694
VDSR [19]	2	665K	612.6G	37.53 / 0.9587	33.03 / 0.9124	31.90 / 0.8960	37.22 / 0.9729
DRCN [20]	2	1,774K	9,788.7G	37.63 / 0.9588	33.04 / 0.9118	31.85 / 0.8942	37.63 / 0.9723
LapSRN [22]	2	813K	29.9G	37.52 / 0.9590	33.08 / 0.9130	31.80 / 0.8950	37.27 / 0.9740
DRRN [35]	2	297K	6,796.9G	37.74 / 0.9591	33.23 / 0.9136	32.05 / 0.8973	37.92 / 0.9760
MemNet [36]	2	677K	623.9G	37.78 / 0.9597	33.28 / 0.9142	32.08 / 0.8978	/
FALSR-A [6]	2	1,021K	234.7G	37.82 / 0.9595	33.55 / 0.9168	32.12 / 0.8987	/
FALSR-B [6]	2	326K	74.7G	37.61 / 0.9585	33.29 / 0.9143	31.97 / 0.8967	/
FALSR-C [6]	2	408K	93.7G	37.66 / 0.9586	33.26 / 0.9140	31.96 / 0.8965	/
CARN [2]	2	1,592K	222.8G	37.76 / 0.9590	33.52 / 0.9166	32.09 / 0.8978	/
SRMDNF [45]	2	1,513K	347.7G	37.79 / 0.9600	33.32 / 0.9150	32.05 / 0.8980	/
MSRN [28]	2	5,930K	1,365.4G	38.08 / 0.9607	33.70 / 0.9186	32.23 / 0.9002	38.69 / 0.9772
AWSRN [39]	2	1,397K	320.5G	38.11 / 0.9608	33.78 / 0.9189	32.26 / 0.9006	38.87 / 0.9776
FC^{2}N [Ours]	2	1,277K	294.0G	38.11 / 0.9608	33.70 / 0.9179	32.21 / 0.9001	38.81 / 0.9775
FC^{2}N$^+$ [Ours]	2	1,277K	294.0G	38.16 / 0.9610	33.78 / 0.9189	32.25 / 0.9006	39.02 / 0.9779
FC^{2}N^{++} [Ours]	2	1,277K	294.0G	38.17 / 0.9611	33.79 / 0.9191	32.28 / 0.9007	39.04 / 0.9780
Bicubic	3	/	/	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	26.95 / 0.8556
SRCNN [7]	3	57K	52.7G	32.75 / 0.9090	29.28 / 0.8209	28.41 / 0.7863	30.59 / 0.9107
FSRCNN [8]	3	12K	5.0G	33.16 / 0.9140	29.43 / 0.8242	28.53 / 0.7910	30.98 / 0.9212
VDSR [19]	3	665K	612.6G	33.66 / 0.9213	29.77 / 0.8314	28.82 / 0.7976	32.01 / 0.9310
DRCN [20]	3	1,774K	9,788.7G	33.82 / 0.9226	29.76 / 0.8311	28.80 / 0.7963	32.31 / 0.9328
DRRN [35]	3	297K	6,796.9G	34.03 / 0.9244	29.96 / 0.8349	28.95 / 0.8004	32.74 / 0.9390
MemNet [36]	3	677K	623.9G	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	/
CARN [2]	3	1,592K	118.8G	34.29 / 0.9255	30.29 / 0.8407	29.06 / 0.8034	/
SRMDNF [45]	3	1,530K	156.3G	34.12 / 0.9250	30.04 / 0.8370	28.97 / 0.8030	/
MSRN [28]	3	6,114K	625.7G	34.46 / 0.9278	30.41 / 0.8437	29.15 / 0.8064	33.67 / 0.9456
AWSRN [39]	3	1,476K	150.6G	34.52 / 0.9281	30.38 / 0.8426	29.16 / 0.8069	33.85 / 0.9463
FC^{2}N [Ours]	3	1,323K	135.8G	34.53 / 0.9282	30.44 / 0.8437	29.16 / 0.8068	33.83 / 0.9462
FC^{2}N$^+$ [Ours]	3	1,323K	135.8G	34.60 / 0.9287	30.51 / 0.8449	29.20 / 0.8074	34.11 / 0.9476
FC^{2}N^{++} [Ours]	3	1,323K	135.8G	34.61 / 0.9287	30.52 / 0.8450	29.21 / 0.8075	34.16 / 0.9478
Bicubic	4	/	/	28.42 / 0.8104	26.00 / 0.7027	25.96 / 0.6675	24.89 / 0.7866
SRCNN [7]	4	57K	52.7G	30.48 / 0.8628	27.49 / 0.7503	26.90 / 0.7101	27.66 / 0.8505
FSRCNN [8]	4	12K	4.6G	30.71 / 0.8657	27.59 / 0.7535	26.98 / 0.7150	27.90 / 0.8517
VDSR [19]	4	665K	612.6G	31.35 / 0.8838	28.01 / 0.7674	27.29 / 0.7251	28.83 / 0.8809
DRCN [20]	4	1,774K	9,788.7G	31.53 / 0.8854	28.02 / 0.7670	27.23 / 0.7233	28.98 / 0.8816
LapSRN [22]	4	813K	149.4G	31.54 / 0.8850	28.19 / 0.7720	27.32 / 0.7280	29.09 / 0.8845
DRRN [35]	4	297K	6,796.9G	31.68 / 0.8888	28.21 / 0.7720	27.38 / 0.7284	29.46 / 0.8960
MemNet [36]	4	677K	623.9G	31.74 / 0.8893	28.26 / 0.7723	27.40 / 0.7281	/
CARN [2]	4	1,592K	90.9G	32.13 / 0.8937	28.60 / 0.7806	27.58 / 0.7349	/
SRMDNF [45]	4	1,555K	89.3G	31.96 / 0.8930	28.35 / 0.7770	27.49 / 0.7340	/
MSRN [28]	4	6,078K	349.8G	32.26 / 0.8960	28.63 / 0.7836	27.61 / 0.7380	30.57 / 0.9103
AWSRN [39]	4	1,587K	91.1G	32.27 / 0.8960	28.69 / 0.7843	27.64 / 0.7385	30.72 / 0.9109
FC^{2}N [Ours]	4	1,314K	82.6G	32.23 / 0.8956	28.68 / 0.7836	27.62 / 0.7377	30.74 / 0.9110
FC^{2}N$^+$ [Ours]	4	1,314K	82.6G	32.36 / 0.8970	28.75 / 0.7851	27.68 / 0.7390	31.04 / 0.9136
FC^{2}N^{++} [Ours]	4	1,314K	82.6G	32.37 / 0.8971	28.76 / 0.7853	27.68 / 0.7391	31.06 / 0.9137

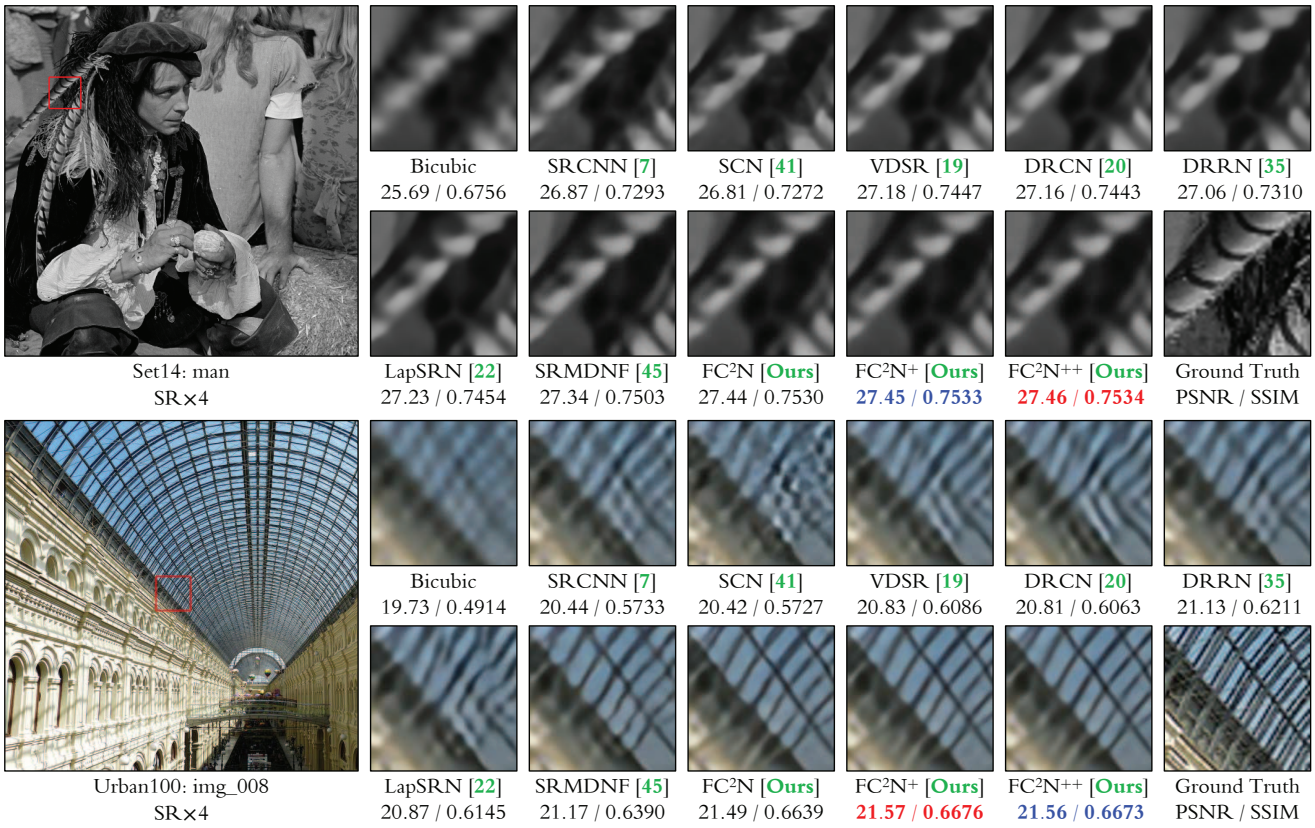


Figure 6. Visual comparison between several typical lightweight SR models and the **lightweight** implementation of our FC^{2}N model with $n = m = 4$. The best and second best results are in red and blue, respectively.

trend to WGFF weights, with relatively small determined values overall (e.g., < 0.6). By comparing λ_1 and λ_2, it can be seen that $\lambda_1 > \lambda_2$ when CG index is greater than 4.

A possible reason for this may be that identity mappings are more conducive to information propagation, especially feedback propagation, than nonlinear mappings. The model thus assigns them more attention to improve the information flow in the network.

Weights in concat block Fig.10 exhibits the learned values for $\{\lambda_1, \lambda_2\}$ in CBs. The model is also with $n = 16, m = 8$ and SR×4. Note that λ_1 is for the identity branch and λ_2 is for the nonlinear mapping branch. At this time, we can more obviously see that λ_1 weights have larger values than λ_2 weights on the whole. Compared with in CGs, weighted channel concatenations in CBs can be regarded as a kind of adaptive "short-term" skip connections, and the selection mechanism seems more obvious in this case.

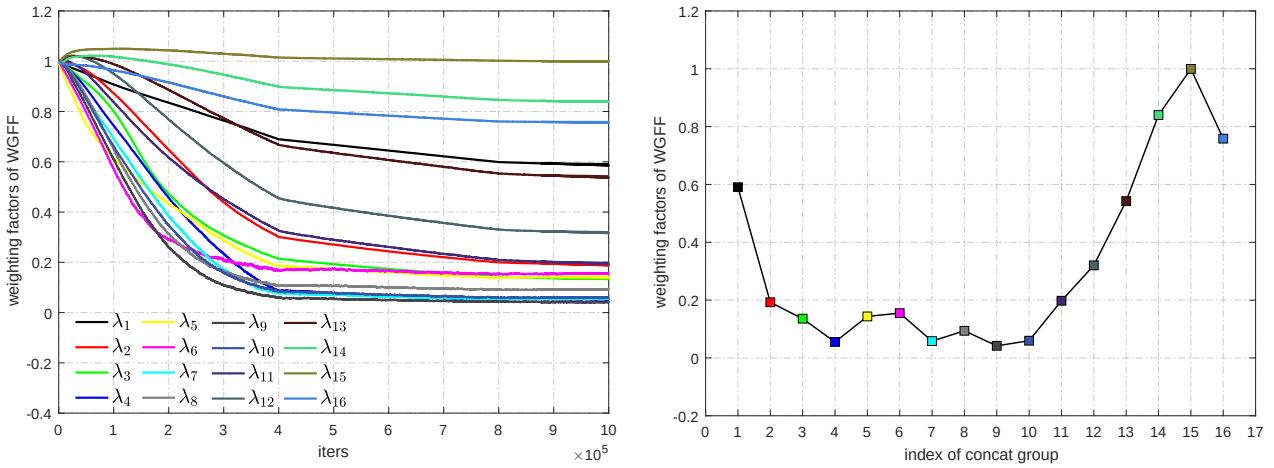
Another important observation in Fig.10 is that at some layers, both λ_1 and λ_2 have small values very close to 0, for instance, the combined index of n and m is 32, 48, 82∼84 and 107∼112 etc. This implies that there are bottlenecks at these locations and information may be propagated through other interlayer connections. Therefore, the network shows some "sparsity" in its interlayer connections, resulting that the "effective" depth of the network may be smaller than its actual depth. This viewpoint is somewhat similar to that of [38], which holds that residual networks can be regarded as the ensembles of multiple shallow networks and they enable very deep network by shortening the *effective* paths.

C. Network scale

Network scale refers to the depth, width and parameter scale of the network. The depth of the network is usually defined as the longest path from the network input to the output, and the width is usually the maximum number of feature channels. Both of them affect the parameter scale of the network. According to the network architecture shown in Fig.2 and Fig.3, the depth of our FC^{2}N is given by:

$$D = n(3m + 1) + 4 + u, \tag{14}$$

where m and n denote the number of CBs and CGs respectively. u is the depth of the upscale module, which depends on the SR scaling factor. Specifically, $u = 1$ for SR×2 and SR×3, $u = 2$ for SR×4, and $u = 3$ for SR×8. The "3" in



(a) WGFF weight evolution

(b) Determined WGFF weights

Figure 7. The evolution of weighting factors in WGFF during the model training (a) and their finally determined values (b). The model is with $n = 16, m = 8$ and SR×4. We exclude the first weighting factor λ_0 and only focus on those for CGs here.

the parentheses corresponds to 3 conv layers in a CB, and "1" in the parentheses refers to the 1×1 conv layer at the end of each CG. The width of FC^{2}N network is determined the width of each CB. We set the feature channels of a CB as $\{32, 128, 32\}$, so the network width can be approximately viewed as 128. In addition, FC^{2}N has fewer network parameters than other advanced methods, e.g., compared with RCAN [46], it reduces model parameters by about 40%, but achieves better SR performance.

D. Analysis on wide activation

In the context of single image SR, low-level information in deep models may contribute to more accurate pixel-wise predication [43, 39]. Wide activation is considered to ease the propagation of low-level features in the network, thus boosting model performance. To verify the effectiveness of wide activation, we compare several configurations of wide activation in this section. Fig.11 shows the validation curves of the lightweight FC^{2}N model ($n = 4, m = 4$) on Set5 [4] with SR×2, which is equipped with different configurations of wide activation.

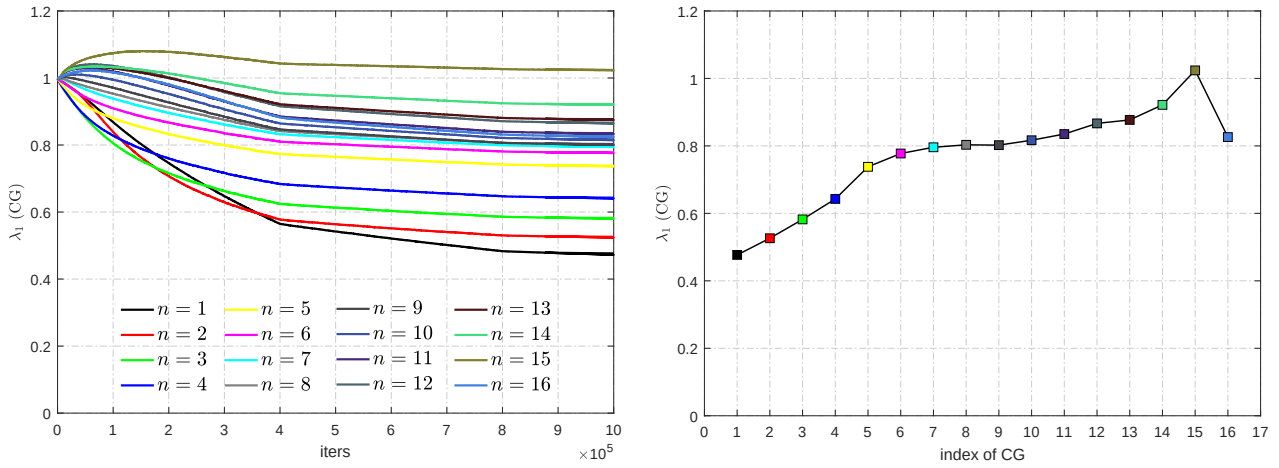
Assume the width of identity mapping pathway is w_1 and the width before the activation inside nonlinear pathway is w_2. Let r_{wa} denote the ratio of w_1 and w_2:

$$r_{wa} = \frac{w_2}{w_1}. \tag{15}$$

For fairness of comparison, we keep the feature width of the feature extraction subnet and image reconstruction subnet the same as FC^{2}N. It can be seen that properly increasing r_{wa} favors to performance improvement, e.g., $r_{wa} = 1$ and $r_{wa} = 4$. However, model performance will degrade as r_{wa} continues to increase and reaches a certain threshold, e.g., $r_{wa} = 64$. Similar phenomenon was also observed in [43] and one possible reason for this performance degradation is that the identity mapping becomes too slim, resulting in the bottleneck of low-level information propagation.

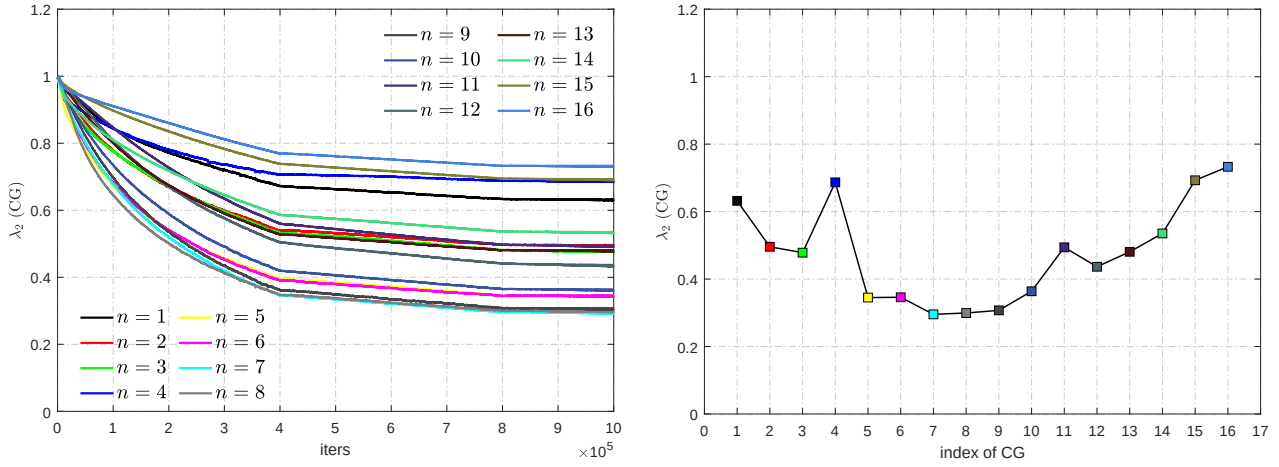
E. Discussions

Residual blocks in EDSR Residual blocks in EDSR [29] are widely applied in various image generation tasks, which are deemed to help more low-level features to pass through while still maintian the highly non-linearity of deep neural networks [46], [14], [50], [43], [39], [2], [47]. Although the skip connections formed by element-wise addition between the nonlinear mapping and identity mapping are conducive to feedback propagation [12], the bypass paths constructed in this way make residual networks behave like ensembles of multiple relatively shallow networks [38]. Besides, the performance gain obtained by most of the residual networks is mainly attributed to *a simple but essential concept – going deeper* [11, 38], such as VDSR [19], EDSR / MDSR [29] and RCAN [46] in term of image SR. This indicates that residual networks may not be conducive to fully exploiting model representational capacity due to their ensemble-like behavior. Unlike the residual blocks in EDSR [29], the skip connection in our CBs is formed by channel concatenation followed by a 1×1 conv layer, which can be viewed as the weighted combination of the nonlinear mapping branch and identity mapping branch. Therefore, compared with basic element-wise addition, the channel concatenation followed by a 1×1 conv layer can integrate the linear and nonlinear features in the network more effectively, further mining the representational capacity of the model. This can be verified by comparing baseline [res] and FC^{2}N-000 in Fig.4. While the non-residual nature of the proposed FC^{2}N model seems



(a) Evolution curves of λ_1 in CGs (b) Determined values of λ_1 in CGs

Figure 8. Evolution curves (a) and the determined values (b) of the weighting factor λ_1 for the **identity** mapping branch. The network is with $n = 16, m = 8$ and SR$\times$4.



(a) Evolution curves of λ_2 in CGs (b) Determined values of λ_2 in CGs

Figure 9. Evolution curves (a) and the determined values (b) of the weighting factor λ_2 for the **nonlinear** mapping branch. The network is with $n = 16, m = 8$ and SR$\times$4.

to be detrimental to feedback propagation according to the derivation of He *et al*. [12], it still reaches the network depth over 400 layers[2]. In fact, according to previous explanation in section 3.1, the basic residual block of EDSR [29] is a special case of our CBs.

Residual blocks in AWSRN The adaptive residual block in AWSRN [39] is a variant of the basic residual block in EDSR [29], in which the nonlinear mapping and identity mapping are weighted by two learnable factors. It is also a special case of our CB block when channel concatenation with 1×1 conv degrades to residual addition. Weighting the branches of the basic residual blocks is mainly inspired by the trick of residual scaling [34, 29], which is typically used to stabilize the model training. It is expected to help extend model representational capacity by adaptively adjusting the ratio between the nonlinear and identity mapping branches of a residual block. According to [39], the deeper building block (i.e., adaptive weighted residual unit, AWRU) in the network requires a smaller weighting factor to prevent from gradient explosion.

Actually, in AWSRN [39], the weighting factors of both identity mapping and nonlinear mapping branches decrease with the increase in network depth. However, similar trend cannot be observed in Fig.10 in term of the proposed FC^{2}N

[2]To the best of our knowledge, RCAN [46] is probably the deepest SR network at present that is also over 400 layers. However, it is a kind of residual networks.

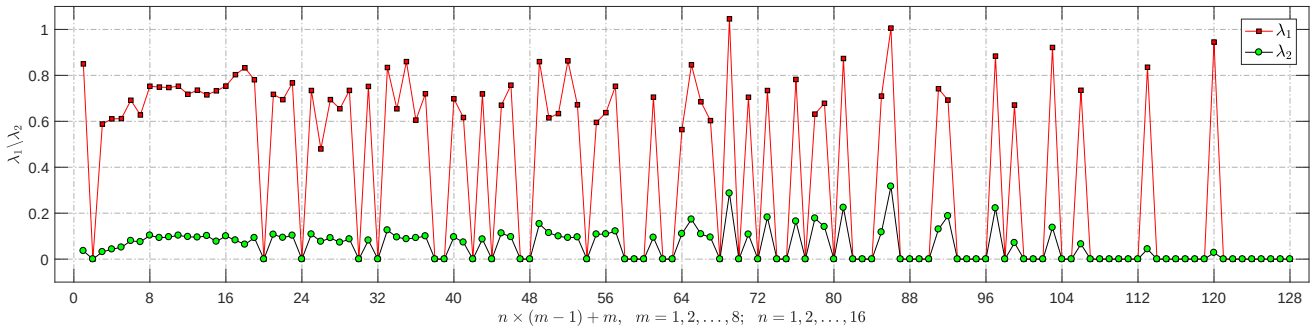


Figure 10. Weighting factors determined by model training in CBs (FC^{2}N with $n = 16, m = 8$ and SR$\times$4). The x-coordinate denotes the combined index of m and n that reflects the corresponding network depth.

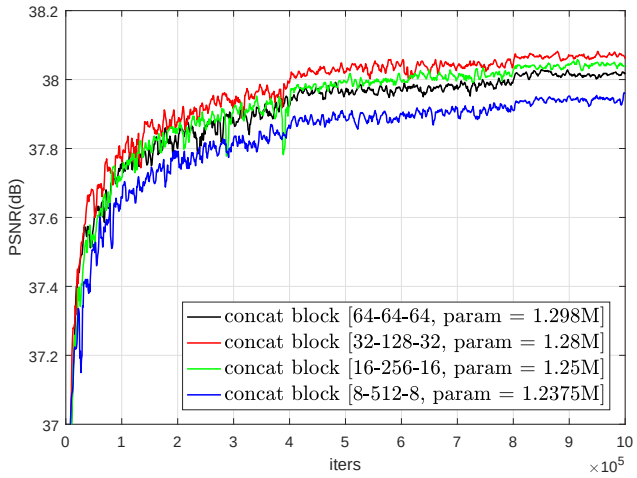


Figure 11. Validation curves of FC^{2}N ($n = m = 4$) with different configurations of wide activation. The results are collected on Set5 [4] with SR$\times$2.

Table 5. Testing results corresponding to different configurations of wide activation shown in Fig.11. These results are calculated on Set5 [4] for 3 typical scaling factors (PSNR (dB) / SSIM).

CB config	SR$\times$2	SR$\times$3	SR$\times$4
64–64–64	38.04 / 0.960	34.48 / 0.926	32.18 / 0.895
32–128–32	38.11 / 0.961	34.53 / 0.928	32.23 / 0.896
16–256–16	38.10 / 0.961	34.51 / 0.928	32.24 / 0.896
8–512–8	37.97 / 0.958	34.41 / 0.925	32.15 / 0.895

model. This indicates that the weighting factors in our CBs work in a different manner from that in AWSRN [39]. In our FC^{2}N model, the role of these weighting factors seems to be more in the fusion of linear and nonlinear features in the model and the selection of information propagation paths across the entire network, rather than in residual scaling to prevent gradient explosion.

Attention Mechanism The target of attention mechanism in deep neural networks is to retune the feature responses towards the most informative and important components of the inputs [13, 46, 50, 14]. In implementation, it is typically combined with a gating function used for nonlinearity, e.g., a softmax or sigmoid [13]. In the context of image SR, it mainly refers to channel attention (also known as channel discrimination), e.g., RCAN [46] and CSN [50], and spatial attention, e.g., CSFM [14]. However, most of these works are based on self-attention mechanism, i.e., adding attention modules to hierarchical features itself, e.g., residual channel attention block (RCAB) in RCAN [46] is implemented by adding an attention module at the tail of the basic residual block of EDSR [29], where the attention module includes a sigmoid function and sequential operations. The proposed concat block can also realize attention mechanism but in a different manner: (1) the 1$\times$1 conv layer allows the model to learn *linear* interactions between different channels but the channel-wise features are still emphasized opposed to one-hot activation; (2) joint attention of linear and nonlinear features is achieved by channel concatenation, instead of the self-attention only on nonlinear features.

F. More visual comparisons

We present more results of visual comparison to further illustrate the advantages of the proposed FC^{2}N model over other SR approaches (Fig.12 $\sim$ Fig.14).

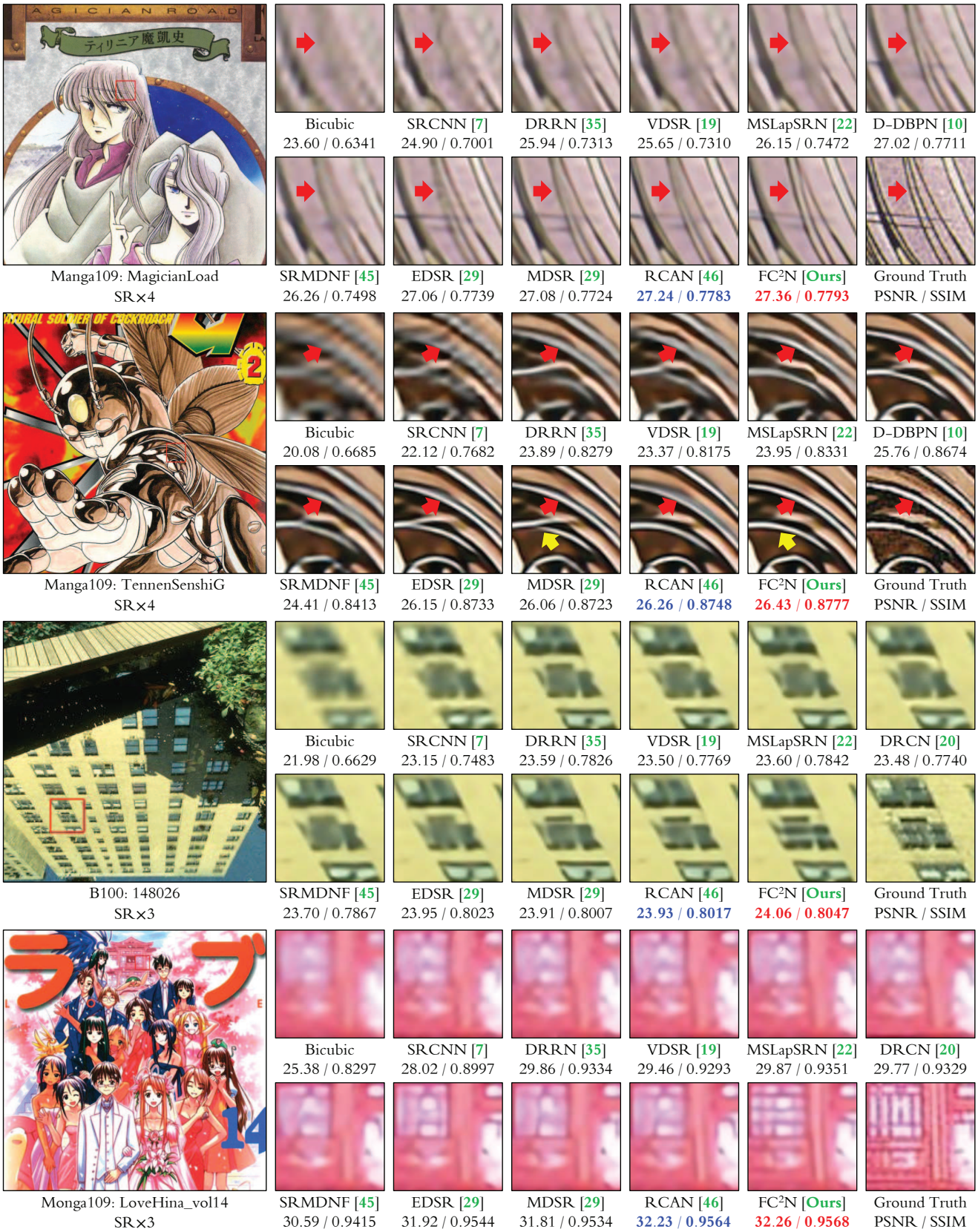


Figure 12. Visual comparison between several representational SR methods and our FC^{2}N model with $n = 16, m = 8$. The best and second best results are in red and blue, respectively.

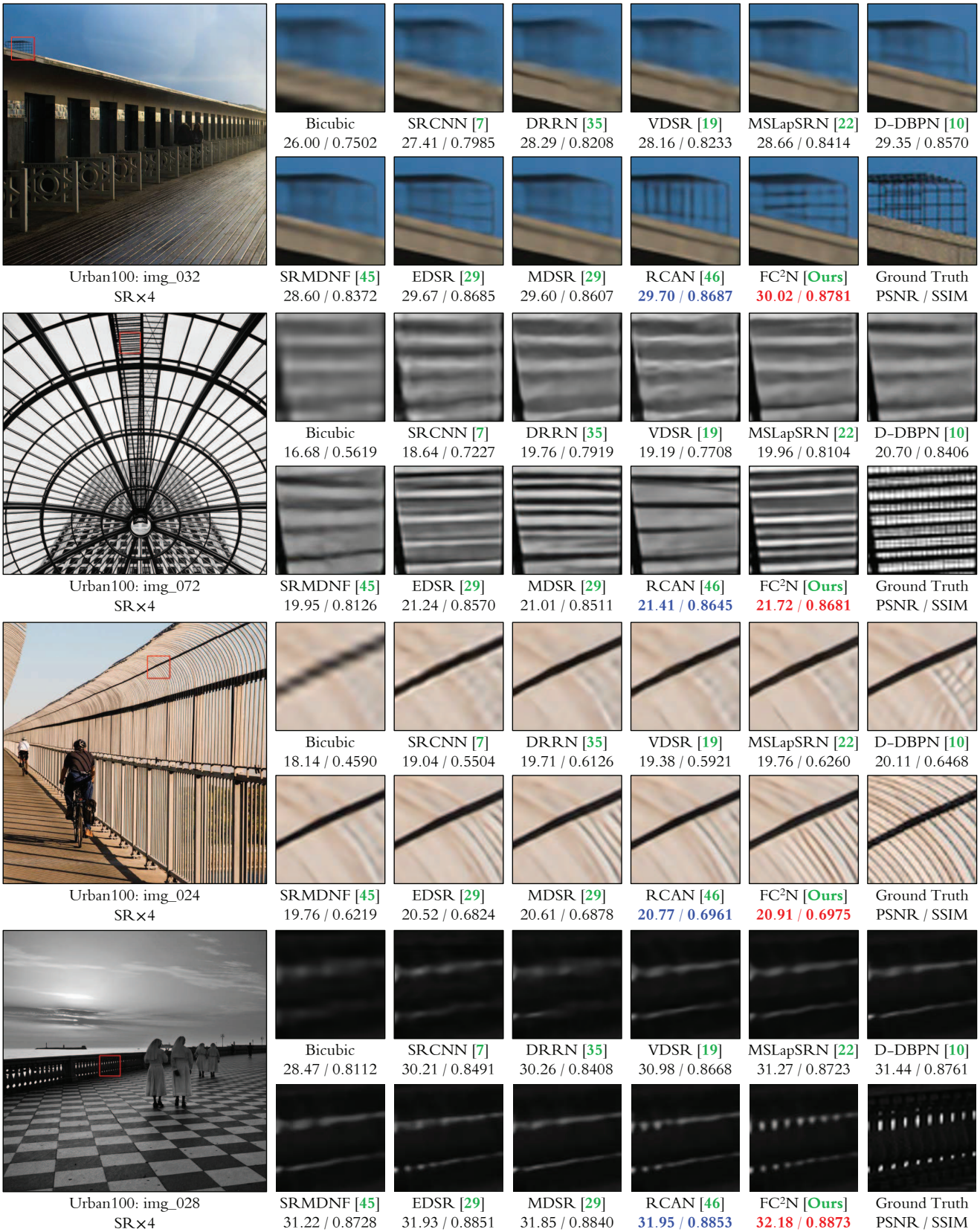


Figure 13. Visual comparison between several representational SR methods and our FC^{2}N model with $n = 16, m = 8$. The best and second best results are in red and blue, respectively.

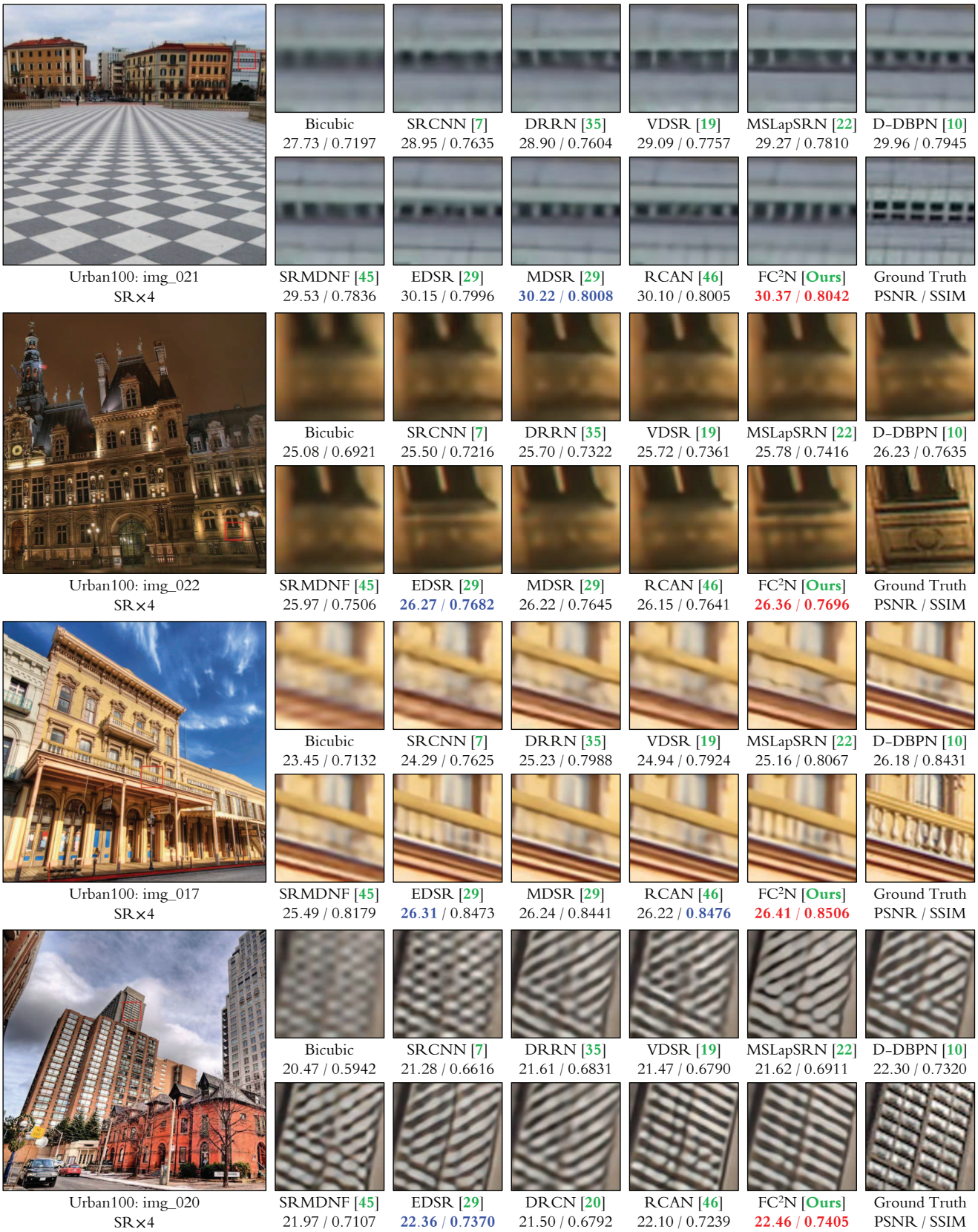


Figure 14. Visual comparison between several representational SR methods and our FC^{2}N model with $n = 16, m = 8$. The best and second best results are in red and blue, respectively.