Residual Dense Network for Image Super-Resolution

Yulun Zhang[1], Yapeng Tian[2], Yu Kong[1], Bineng Zhong[1], Yun Fu[1,3]
[1]Department of Electrical and Computer Engineering, Northeastern University, Boston, USA
[2]Department of Computer Science, University of Rochester, Rochester, USA
[3]College of Computer and Information Science, Northeastern University, Boston, USA
yulun100@gmail.com, yapengtian@rochester.edu, bnzhong@hqu.edu.cn, {yukong,yunfu}@ece.neu.edu

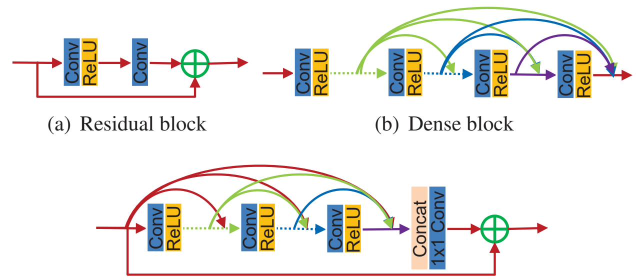
Abstract

A very deep convolutional neural network (CNN) has recently achieved great success for image super-resolution (SR) and offered hierarchical features as well. However, most deep CNN based SR models do not make full use of the hierarchical features from the original low-resolution (LR) images, thereby achieving relatively-low performance. In this paper, we propose a novel residual dense network (RDN) to address this problem in image SR. We fully exploit the hierarchical features from all the convolutional layers. Specifically, we propose residual dense block (RDB) to extract abundant local features via dense connected convolutional layers. RDB further allows direct connections from the state of preceding RDB to all the layers of current RDB, leading to a contiguous memory (CM) mechanism. Local feature fusion in RDB is then used to adaptively learn more effective features from preceding and current local features and stabilizes the training of wider network. After fully obtaining dense local features, we use global feature fusion to jointly and adaptively learn global hierarchical features in a holistic way. Experiments on benchmark datasets with different degradation models show that our RDN achieves favorable performance against state-of-the-art methods.

1. Introduction

Single image Super-Resolution (SISR) aims to generate a visually pleasing high-resolution (HR) image from its degraded low-resolution (LR) measurement. SISR is used in various computer vision tasks, such as security and surveillance imaging [42], medical imaging [23], and image generation [9]. While image SR is an ill-posed inverse procedure, since there exists a multitude of solutions for any LR input. To tackle this inverse problem, plenty of image SR algorithms have been proposed, including interpolation-based [40], reconstruction-based [37], and learning-based methods [28, 29, 20, 2, 21, 8, 10, 31, 39].

Among them, Dong et al. [2] firstly introduced a three-



(a) Residual block (b) Dense block

(c) Residual dense block

Figure 1. Comparison of prior network structures (a,b) and our residual dense block (c). (a) Residual block in MDSR [17]. (b) Dense block in SRDenseNet [31]. (c) Our residual dense block.

layer convolutional neural network (CNN) into image SR and achieved significant improvement over conventional methods. Kim et al. increased the network depth in VDSR [10] and DRCN [11] by using gradient clipping, skip connection, or recursive-supervision to ease the difficulty of training deep network. By using effective building modules, the networks for image SR are further made deeper and wider with better performance. Lim et al. used residual blocks (Fig. 1(a)) to build a very wide network EDSR [17] with residual scaling [24] and a very deep one MDSR [17]. Tai et al. proposed memory block to build MemNet [26]. As the network depth grows, the features in each convolutional layer would be hierarchical with different receptive fields. However, these methods neglect to fully use information of each convolutional layer. Although the gate unit in memory block was proposed to control short-term memory [26], the local convolutional layers don't have direct access to the subsequent layers. So it's hard to say memory block makes full use of the information from all the layers within it.

Furthermore, objects in images have different scales, angles of view, and aspect ratios. Hierarchical features from a very deep network would give more clues for reconstruction. While, most deep learning (DL) based methods (e.g., VDSR [10], LapSRN [13], and EDSR [17]) neglect to use hierarchical features for reconstruction. Although memory block [26] also takes information from preceding memory blocks as input, the multi-level features are not extracted from the original LR image. MemNet interpolates the original LR image to the desired size to form the input. This pre-processing step not only increases computation complexity quadratically, but also loses some details of the original LR image. Tong et al. introduced dense block (Fig. 1(b)) for image SR with relatively low growth rate (e.g.,16). According to our experiments (see Section 5.2), higher growth rate can further improve the performance of the network. While, it would be hard to train a wider network with dense blocks in Fig. 1(b).

To address these drawbacks, we propose residual dense network (RDN) (Fig. 2) to fully make use of all the hierarchical features from the original LR image with our proposed residual dense block (Fig. 1(c)). It's hard and impractical for a very deep network to directly extract the output of each convolutional layer in the LR space. We propose residual dense block (RDB) as the building module for RDN. RDB consists dense connected layers and local feature fusion (LFF) with local residual learning (LRL). Our RDB also support contiguous memory among RDBs. The output of one RDB has direct access to each layer of the next RDB, resulting in a contiguous state pass. Each convolutional layer in RDB has access to all the subsequent layers and passes on information that needs to be preserved [7]. Concatenating the states of preceding RDB and all the preceding layers within the current RDB, LFF extracts local dense feature by adaptively preserving the information. Moreover, LFF allows very high growth rate by stabilizing the training of wider network. After extracting multi-level local dense features, we further conduct global feature fusion (GFF) to adaptively preserve the hierarchical features in a global way. As depicted in Figs. 2 and 3, each layer has direct access to the original LR input, leading to an implicit deep supervision [15].

In summary, our main contributions are three-fold:

- We propose a unified frame work residual dense network (RDN) for high-quality image SR with different degradation models. The network makes full use of all the hierarchical features from the original LR image.
- We propose residual dense block (RDB), which can not only read state from the preceding RDB via a contiguous memory (CM) mechanism, but also fully utilize all the layers within it via local dense connections. The accumulated features are then adaptively preserved by local feature fusion (LFF).
- We propose global feature fusion to adaptively fuse hierarchical features from all RDBs in the LR space. With global residual learning, we combine the shallow features and deep features together, resulting in global dense features from the original LR image.

2. Related Work

Recently, deep learning (DL)-based methods have achieved dramatic advantages against conventional methods in computer vision [36, 35, 34, 16]. Due to the limited space, we only discuss some works on image SR. Dong et al. proposed SRCNN [2], establishing an end-to-end mapping between the interpolated LR images and their HR counterparts for the first time. This baseline was then further improved mainly by increasing network depth or sharing network weights. VDSR [10] and IRCNN [38] increased the network depth by stacking more convolutional layers with residual learning. DRCN [11] firstly introduced recursive learning in a very deep network for parameter sharing. Tai et al. introduced recursive blocks in DRRN [25] and memory block in Memnet [26] for deeper networks. All of these methods need to interpolate the original LR images to the desired size before applying them into the networks. This pre-processing step not only increases computation complexity quadratically [4], but also over-smooths and blurs the original LR image, from which some details are lost. As a result, these methods extract features from the interpolated LR images, failing to establish an end-to-end mapping from the original LR to HR images.

To solve the problem above, Dong et al. [4] directly took the original LR image as input and introduced a transposed convolution layer (also known as deconvolution layer) for upsampling to the fine resolution. Shi et al. proposed ESPCN [22], where an efficient sub-pixel convolution layer was introduced to upscale the final LR feature maps into the HR output. The efficient sub-pixel convolution layer was then adopted in SRResNet [14] and EDSR [17], which took advantage of residual leanrning [6]. All of these methods extracted features in the LR space and upscaled the final LR features with transposed or sub-pixel convolution layer. By doing so, these networks can either be capable of real-time SR (e.g., FSRCNN and ESPCN), or be built to be very deep/wide (e.g., SRResNet and EDSR). However, all of these methods stack building modules (e.g., Conv layer in FSRCNN, residual block in SRResNet and EDSR) in a chain way. They neglect to adequately utilize information from each Conv layer and only adopt CNN features from the last Conv layer in LR space for upscaling.

Recently, Huang et al. proposed DenseNet, which allows direct connections between any two layers within the same dense block [7]. With the local dense connections, each layer reads information from all the preceding layers within the same dense block. The dense connection was introduced among memory blocks [26] and dense blocks [31]. More differences between DenseNet/SRDenseNet/MemNet and our RDN would be discussed in Section 4.

The aforementioned DL-based image SR methods have achieved significant improvement over conventional SR methods, but all of them lose some useful hierarchical fea-

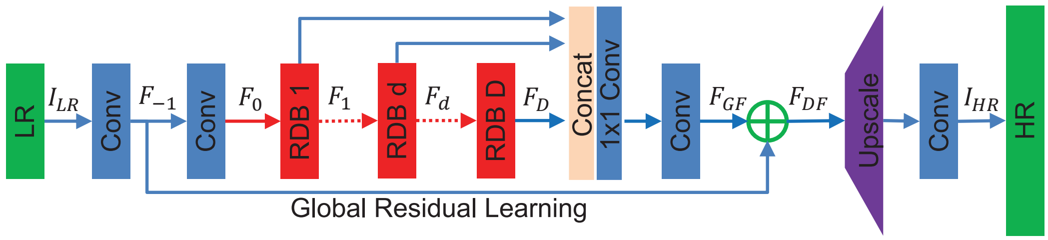


Figure 2. The architecture of our proposed residual dense network (RDN).

tures from the original LR image. Hierarchical features produced by a very deep network are useful for image restoration tasks (e.g., image SR). To fix this case, we propose residual dense network (RDN) to extract and adaptively fuse features from all the layers in the LR space efficiently. We will detail our RDN in next section.

3. Residual Dense Network for Image SR

3.1. Network Structure

As shown in Fig. 2, our RDN mainly consists four parts: shallow feature extraction net (SFENet), redidual dense blocks (RDBs), dense feature fusion (DFF), and finally the up-sampling net (UPNet). Let's denote I_{LR} and I_{SR} as the input and output of RDN. Specifically, we use two Conv layers to extract shallow features. The first Conv layer extracts features F_{-1} from the LR input.

$$F_{-1} = H_{SFE1}\left(I_{LR}\right), \tag{1}$$

where $H_{SFE1}\left(\cdot\right)$ denotes convolution operation. F_{-1} is then used for further shallow feature extraction and global residual learning. So we can further have

$$F_0 = H_{SFE2}\left(F_{-1}\right), \tag{2}$$

where $H_{SFE2}\left(\cdot\right)$ denotes convolution operation of the second shallow feature extraction layer and is used as input to residual dense blocks. Supposing we have D residual dense blocks, the output F_d of the d-th RDB can be obtained by

$$\begin{aligned} F_d &= H_{RDB,d}\left(F_{d-1}\right) \\ &= H_{RDB,d}\left(H_{RDB,d-1}\left(\cdots\left(H_{RDB,1}\left(F_0\right)\right)\cdots\right)\right), \end{aligned} \tag{3}$$

where $H_{RDB,d}$ denotes the operations of the d-th RDB. $H_{RDB,d}$ can be a composite function of operations, such as convolution and rectified linear units (ReLU) [5]. As F_d is produced by the d-th RDB fully utilizing each convolutional layers within the block, we can view F_d as local feature. More details about RDB will be given in Section 3.2.

After extracting hierarchical features with a set of RDBs, we further conduct dense feature fusion (DFF), which includes global feature fusion (GFF) and global residual learning (GRL). DFF makes full use of features from all the preceding layers and can be represented as

$$F_{DF} = H_{DFF}\left(F_{-1}, F_0, F_1, \cdots, F_D\right), \tag{4}$$

where F_{DF} is the output feature-maps of DFF by utilizing a composite function H_{DFF}. More details about DFF will be shown in Section 3.3.

After extracting local and global features in the LR space, we stack a up-sampling net (UPNet) in the HR space. Inspired by [17], we utilize ESPCN [22] in UPNet followed by one Conv layer. The output of RDN can be obtained by

$$I_{SR} = H_{RDN}\left(I_{LR}\right), \tag{5}$$

where H_{RDN} denotes the function of our RDN.

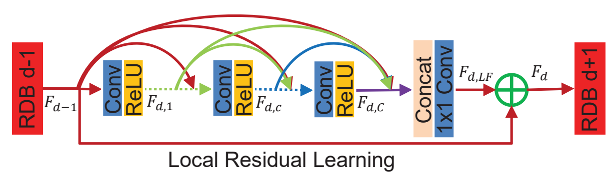


Figure 3. Residual dense block (RDB) architecture.

3.2. Residual Dense Block

Now we present details about our proposed residual dense block (RDB) in Fig. 3. Our RDB contains dense connected layers, local feature fusion (LFF), and local residual learning, leading to a contiguous memory (CM) mechanism.

Contiguous memory mechanism is realized by passing the state of preceding RDB to each layer of current RDB. Let F_{d-1} and F_d be the input and output of the d-th RDB respectively and both of them have G_0 feature-maps. The output of c-th Conv layer of d-th RDB can be formulated as

$$F_{d,c} = \sigma\left(W_{d,c}\left[F_{d-1}, F_{d,1}, \cdots, F_{d,c-1}\right]\right), \tag{6}$$

where σ denotes the ReLU [5] activation function. $W_{d,c}$ is the weights of the c-th Conv layer, where the bias term is omitted for simplicity. We assume $F_{d,c}$ consists of G (also known as growth rate [7]) feature-maps. $[F_{d-1}, F_{d,1}, \cdots, F_{d,c-1}]$ refers to the concatenation of the feature-maps produced by the $(d-1)$-th RDB, convolutional layers $1, \cdots, (c-1)$ in the d-th RDB, resulting in

$G_0+(c-1)\times G$ feature-maps. The outputs of the preceding RDB and each layer have direct connections to all subsequent layers, which not only preserves the feed-forward nature, but also extracts local dense feature.

Local feature fusion is then applied to adaptively fuse the states from preceding RDB and the whole Conv layers in current RDB. As analyzed above, the feature-maps of the $(d-1)$-th RDB are introduced directly to the d-th RDB in a concatenation way, it is essential to reduce the feature number. On the other hand, inspired by MemNet [26], we introduce a 1×1 convolutional layer to adaptively control the output information. We name this operation as local feature fusion (LFF) formulated as

$$F_{d,LF} = H_{LFF}^{d}\left(\left[F_{d-1}, F_{d,1}, \cdots, F_{d,c}, \cdots, F_{d,C}\right]\right), \quad (7)$$

where H_{LFF}^{d} denotes the function of the 1×1 Conv layer in the d-th RDB. We also find that as the growth rate G becomes larger, very deep dense network without LFF would be hard to train.

Local residual learning is introduced in RDB to further improve the information flow, as there are several convolutional layers in one RDB. The final output of the d-th RDB can be obtained by

$$F_d = F_{d-1} + F_{d,LF}. \quad (8)$$

It should be noted that LRL can also further improve the network representation ability, resulting better performance. We introduce more results about LRL in Section 5. Because of the dense connectivity and local residual learning, we refer to this block architecture as residual dense block (RDB). More differences between RDB and original dense block [7] would be summarized in Section 4.

3.3. Dense Feature Fusion

After extracting local dense features with a set of RDBs, we further propose dense feature fusion (DFF) to exploit hierarchical features in a global way. Our DFF consists of global feature fusion (GFF) and global residual learning.

Global feature fusion is proposed to extract the global feature F_{GF} by fusing features from all the RDBs

$$F_{GF} = H_{GFF}\left(\left[F_1, \cdots, F_D\right]\right), \quad (9)$$

where $[F_1, \cdots, F_D]$ refers to the concatenation of feature-maps produced by residual dense blocks $1, \cdots, D$. H_{GFF} is a composite function of 1×1 and 3×3 convolution. The 1×1 convolutional layer is used to adaptively fuse a range of features with different levels. The following 3×3 convolutional layer is introduced to further extract features for global residual learning, which has been demonstrated to be effective in [14].

Global residual learning is then utilized to obtain the feature-maps before conducting up-scaling by

$$F_{DF} = F_{-1} + F_{GF}, \quad (10)$$

where F_{-1} denotes the shallow feature-maps. All the other layers before global feature fusion are fully utilized with our proposed residual dense blocks (RDBs). RDBs produce multi-level local dense features, which are further adaptively fused to form F_{GF}. After global residual learning, we obtain dense feature F_{DF}.

It should be noted that Tai et al. [26] utilized long-term dense connections in MemNet to recover more high frequency information. However, in the memory block [26], the preceding layers don't have direct access to all the subsequent layers. The local feature information are not fully used, limiting the ability of long-term connections. In addition, MemNet extracts features in the HR space, increasing computational complexity. While, inspired by [4, 22, 13, 17], we extract local and global features in the LR space. More differences between our residual dense network and MemNet would be shown in Section 4. We would also demonstrate the effectiveness of global feature fusion in Section 5.

3.4. Implementation Details

In our proposed RDN, we set 3×3 as the size of all convolutional layers except that in local and global feature fusion, whose kernel size is 1×1. For convolutional layer with kernel size 3×3, we pad zeros to each side of the input to keep size fixed. Shallow feature extraction layers, local and global feature fusion layers have G_0=64 filters. Other layers in each RDB has G filters and are followed by ReLU [5]. Following [17], we use ESPCNN [22] to upscale the coarse resolution features to fine ones for the UPNet. The final Conv layer has 3 output channels, as we output color HR images. However, the network can also process gray images.

4. Discussions

Difference to DenseNet. Inspired from DenseNet [7], we adopt the local dense connections into our proposed residual dense block (RDB). In general, DenseNet is widely used in high-level computer vision tasks (e.g., object recognition). While RDN is designed for image SR. Moreover, we remove batch nomalization (BN) layers, which consume the same amount of GPU memory as convolutional layers, increase computational complexity, and hinder performance of the network. We also remove the pooling layers, which could discard some pixel-level information. Furthermore, transition layers are placed into two adjacent dense blocks in DenseNet. While in RDN, we combine dense connected layers with local feature fusion (LFF) by using local residual learning, which would be demonstrated to be effective in Section 5. As a result, the output of the $(d-1)$-th RDB has direct connections to each layer in the d-th RDB and also contributes to the input of $(d+1)$-th RDB. Last not the least, we adopt global feature fusion to fully use hierarchical features, which are neglected in DenseNet.

Difference to SRDenseNet. There are three main differences between SRDenseNet [31] and our RDN. The first one is the design of basic building block. SRDenseNet introduces the basic dense block from DenseNet [7]. Our residual dense block (RDB) improves it in three ways: (1). We introduce contiguous memory (CM) mechanism, which allows the state of preceding RDB have direct access to each layer of the current RDB. (2). Our RDB allow larger growth rate by using local feature fusion (LFF), which stabilizes the training of wide network. (3). Local residual learning (LRL) is utilized in RDB to further encourage the flow of information and gradient. The second one is there is no dense connections among RDB. Instead we use global feature fusion (GFF) and global residual learning to extract global features, because our RDBs with contiguous memory have fully extracted features locally. As shown in Sections 5.2 and 5.3, all of these components increase the performance significantly. The third one is SRDenseNet uses L_2 loss function. Whereas we utilize L_1 loss function, which has been demonstrated to be more powerful for performance and convergence [17]. As a result, our proposed RDN achieves better performance than that of SRDenseNet.

Difference to MemNet. In addition to the different choice of loss function (L_2 in MemNet [26]), we mainly summarize another three differences bwtween MemNet and our RDN. First, MemNet needs to upsample the original LR image to the desired size using Bicubic interpolation. This procedure results in feature extraction and reconstruction in HR space. While, RDN extracts hierarchical features from the original LR image, reducing computational complexity significantly and improving the performance. Second, the memory block in MemNet contains recursive and gate units. Most layers within one recursive unit don't receive the information from their preceding layers or memory block. While, in our proposed RDN, the output of RDB has direct access to each layer of the next RDB. Also the information of each convolutional layer flow into all the subsequent layers within one RDB. Furthermore, local residual learning in RDB improves the flow of information and gradients and performance, which is demonstrated in Section 5. Third, as analyzed above, current memory block doesn't fully make use of the information of the output of the preceding block and its layers. Even though MemNet adopts densely connections among memory blocks in the HR space, MemNet fails to fully extract hierarchical features from the original LR inputs. While, after extracting local dense features with RDBs, our RDN further fuses the hierarchical features from the whole preceding layers in a global way in the LR space.

5. Experimental Results

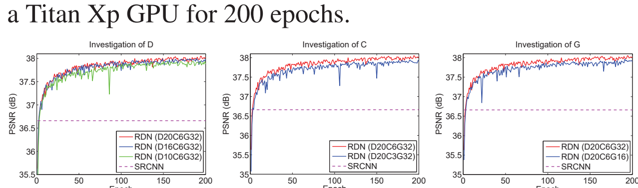
The source code of the proposed method can be downloaded at https://github.com/yulunzhang/RDN.

5.1. Settings

Datasets and Metrics. Recently, Timofte et al. have released a high-quality (2K resolution) dataset DIV2K for image restoration applications [27]. DIV2K consists of 800 training images, 100 validation images, and 100 test images. We train all of our models with 800 training images and use 5 validation images in the training process. For testing, we use five standard benchmark datasets: Set5 [1], Set14 [33], B100 [18], Urban100 [8], and Manga109 [19]. The SR results are evaluated with PSNR and SSIM [32] on Y channel (*i.e.*, luminance) of transformed YCbCr space.

Degradation Models. In order to fully demonstrate the effectiveness of our proposed RDN, we use three degradation models to simulate LR images. The first one is bicubic downsampling by adopting the Matlab function *imresize* with the option *bicubic* (denote as **BI** for short). We use **BI** model to simulate LR images with scaling factor $\times 2$, $\times 3$, and $\times 4$. Similar to [38], the second one is to blur HR image by Gaussian kernel of size 7×7 with standard deviation 1.6. The blurred image is then downsampled with scaling factor $\times 3$ (denote as **BD** for short). We further produce LR image in a more challenging way. We first bicubic downsample HR image with scaling factor $\times 3$ and then add Gaussian noise with noise level 30 (denote as **DN** for short).

Training Setting. Following settings of [17], in each training batch, we randomly extract 16 LR RGB patches with the size of 32×32 as inputs. We randomly augment the patches by flipping horizontally or vertically and rotating 90°. 1,000 iterations of back-propagation constitute an epoch. We implement our RDN with the Torch7 framework and update it with Adam optimizer [12]. The learning rate is initialized to 10^{-4} for all layers and decreases half for every 200 epochs. Training a RDN roughly takes 1 day with a Titan Xp GPU for 200 epochs.



(a) (b) (c)

Figure 4. Convergence analysis of RDN with different values of D, C, and G.

5.2. Study of D, C, and G.

In this subsection, we investigate the basic network parameters: the number of RDB (denote as D for short), the number of Conv layers per RDB (denote as C for short), and the growth rate (denote as G for short). We use the performance of SRCNN [3] as a reference. As shown in Figs. 4(a) and 4(b), larger D or C would lead to higher performance.

	Different combinations of CM, LRL, and GFF							
CM	×	✓	×	×	✓	✓	×	✓
LRL	×	×	✓	×	✓	×	✓	✓
GFF	×	×	×	✓	×	✓	✓	✓
PSNR	34.87	37.89	37.92	37.78	37.99	37.98	37.97	38.06

Table 1. Ablation investigation of contiguous memory (CM), local residual learning (LRL), and global feature fusion (GFF). We observe the best performance (PSNR) on Set5 with scaling factor $\times 2$ in 200 epochs.

This is mainly because the network becomes deeper with larger D or C. As our proposed LFF allows larger G, we also observe larger G (see Fig. 4(c)) contributes to better performance. On the other hand, RND with smaller D, C, or G would suffer some performance drop in the training, but RDN would still outperform SRCNN [3]. More important, our RDN allows deeper and wider network, from which more hierarchical features are extracted for higher performance.

5.3. Ablation Investigation

Table 1 shows the ablation investigation on the effects of contiguous memory (CM), local residual learning (LRL), and global feature fusion (GFF). The eight networks have the same RDB number (D = 20), Conv number (C = 6) per RDB, and growth rate (G = 32). We find that local feature fusion (LFF) is needed to train these networks properly, so LFF isn't removed by default. The baseline (denote as RDN_CM0LRL0GFF0) is obtained without CM, LRL, or GFF and performs very poorly (PSNR = 34.87 dB). This is caused by the difficulty of training [3] and also demonstrates that stacking many basic dense blocks [7] in a very deep network would not result in better performance.

We then add one of CM, LRL, or GFF to the baseline, resulting in RDN_CM1LRL0GFF0, RDN_CM0LRL1GFF0, and RDN_CM0LRL0GFF1 respectively (from 2^{nd} to 4^{th} combination in Table 1). We can validate that each component can efficiently improve the performance of the baseline. This is mainly because each component contributes to the flow of information and gradient.

We further add two components to the baseline, resulting in RDN_CM1LRL1GFF0, RDN_CM1LRL0GFF1, and RDN_CM0LRL1GFF1 respectively (from 5^{th} to 7^{th} combination in Table 1). It can be seen that two components would perform better than only one component. Similar phenomenon can be seen when we use these three components simultaneously (denote as RDN_CM1LRL1GFF1). RDN using three components performs the best.

We also visualize the convergence process of these eight combinations in Fig. 5. The convergence curves are consistent with the analyses above and show that CM, LRL, and GFF can further stabilize the training process without obvious performance drop. These quantitative and visual analyses demonstrate the effectiveness and benefits of our proposed CM, LRL, and GFF.

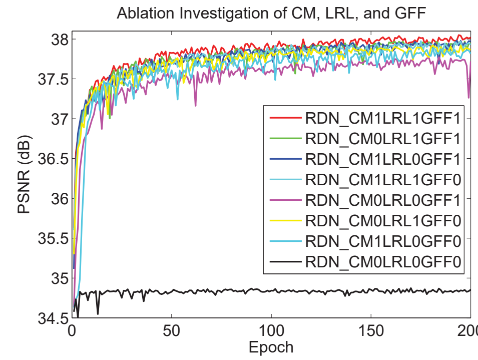


Figure 5. Convergence analysis on CM, LRL, and GFF. The curves for each combination are based on the PSNR on Set5 with scaling factor $\times 2$ in 200 epochs.

5.4. Results with BI Degradation Model

Simulating LR image with BI degradation model is widely used in image SR settings. For BI degradation model, we compare our RDN with 6 state-of-the-art image SR methods: SRCNN [3], LapSRN [13], DRRN [25], SRDenseNet [31], MemNet [26], and MDSR [17]. Similar to [30, 17], we also adopt self-ensemble strategy [17] to further improve our RDN and denote the self-ensembled RDN as RDN+. As analyzed above, a deeper and wider RDN would lead to a better performance. On the other hand, as most methods for comparison only use about 64 filters per Conv layer, we report results of RDN by using D = 16, C = 8, and G = 64 for fair comparison. EDSR [17] is skipped here, because it uses far more filters (*i.e.*, 256) per Conv layer, leading to a very wide network with high number of parameters. However, our RDN would also achieve comparable or even better results than those by EDSR [17].

Table 2 shows quantitative comparisons for $\times 2$, $\times 3$, and $\times 4$ SR. Results of SRDenseNet [31] are cited from their paper. When compared with persistent CNN models (SRDenseNet [31] and MemNet [26]), our RDN performs the best on all datasets with all scaling factors. This indicates the better effectiveness of our residual dense block (RDB) over dense block in SRDensenet [31] and memory block in MemNet [26]. When compared with the remaining models, our RDN also achieves the best average results on most datasets. Specifically, for the scaling factor $\times 2$, our RDN performs the best on all datasets. When the scaling factor becomes larger (e.g., $\times 3$ and $\times 4$), RDN would not hold the similar advantage over MDSR [17]. There are mainly three reasons for this case. First, MDSR is deeper (160 v.s. 128), having about 160 layers to extract features in LR space. Second, MDSR utilizes multi-scale inputs as VDSR does [10]. Third, MDSR uses larger input patch size (65 v.s. 32) for training. As most images in Urban100 contain self-similar structures, larger input patch size for training allows a very deep network to grasp more information by using large receptive field better. As we mainly focus on

Dataset	Scale	Bicubic	SRCNN [3]	LapSRN [13]	DRRN [25]	SRDenseNet [31]	MemNet [26]	MDSR [17]	RDN (ours)	RDN+ (ours)
Set5	×2	33.66/0.9299	36.66/0.9542	37.52/0.9591	37.74/0.9591	-/-	37.78/0.9597	38.11/0.9602	38.24/0.9614	**38.30/0.9616**
	×3	30.39/0.8682	32.75/0.9090	33.82/0.9227	34.03/0.9244	-/-	34.09/0.9248	34.66/0.9280	34.71/0.9296	**34.78/0.9300**
	×4	28.42/0.8104	30.48/0.8628	31.54/0.8855	31.68/0.8888	32.02/0.8934	31.74/0.8893	32.50/0.8973	32.47/0.8990	**32.61/0.9003**
Set14	×2	30.24/0.8688	32.45/0.9067	33.08/0.9130	33.23/0.9136	-/-	33.28/0.9142	33.85/0.9198	34.01/0.9212	**34.10/0.9218**
	×3	27.55/0.7742	29.30/0.8215	29.79/0.8320	29.96/0.8349	-/-	30.00/0.8350	30.44/0.8452	30.57/0.8468	**30.67/0.8482**
	×4	26.00/0.7027	27.50/0.7513	28.19/0.7720	28.21/0.7721	28.50/0.7782	28.26/0.7723	28.72/0.7857	28.81/0.7871	**28.92/0.7893**
B100	×2	29.56/0.8431	31.36/0.8879	31.80/0.8950	32.05/0.8973	-/-	32.08/0.8978	32.29/0.9007	32.34/0.9017	**32.40/0.9022**
	×3	27.21/0.7385	28.41/0.7863	28.82/0.7973	28.95/0.8004	-/-	28.96/0.8001	29.25/0.8091	29.26/0.8093	**29.33/0.8105**
	×4	25.96/0.6675	26.90/0.7101	27.32/0.7280	27.38/0.7284	27.53/0.7337	27.40/0.7281	27.72/0.7418	27.72/0.7419	**27.80/0.7434**
Urban100	×2	26.88/0.8403	29.50/0.8946	30.41/0.9101	31.23/0.9188	-/-	31.31/0.9195	32.84/0.9347	32.89/0.9353	**33.09/0.9368**
	×3	24.46/0.7349	26.24/0.7989	27.07/0.8272	27.53/0.8378	-/-	27.56/0.8376	28.79/0.8655	28.80/0.8653	**29.00/0.8683**
	×4	23.14/0.6577	24.52/0.7221	25.21/0.7553	25.44/0.7638	26.05/0.7819	25.50/0.7630	26.67/0.8041	26.61/0.8028	**26.82/0.8069**
Manga109	×2	30.80/0.9339	35.60/0.9663	37.27/0.9740	37.60/0.9736	-/-	37.72/0.9740	38.96/0.9769	39.18/0.9780	**39.38/0.9784**
	×3	26.95/0.8556	30.48/0.9117	32.19/0.9334	32.42/0.9359	-/-	32.51/0.9369	34.17/0.9473	34.13/0.9484	**34.43/0.9498**
	×4	24.89/0.7866	27.58/0.8555	29.09/0.8893	29.18/0.8914	-/-	29.42/0.8942	31.11/0.9148	31.00/0.9151	**31.39/0.9184**

Table 2. Benchmark results with **BI** degradation model. Average PSNR/SSIM values for scaling factor ×2, ×3, and ×4.

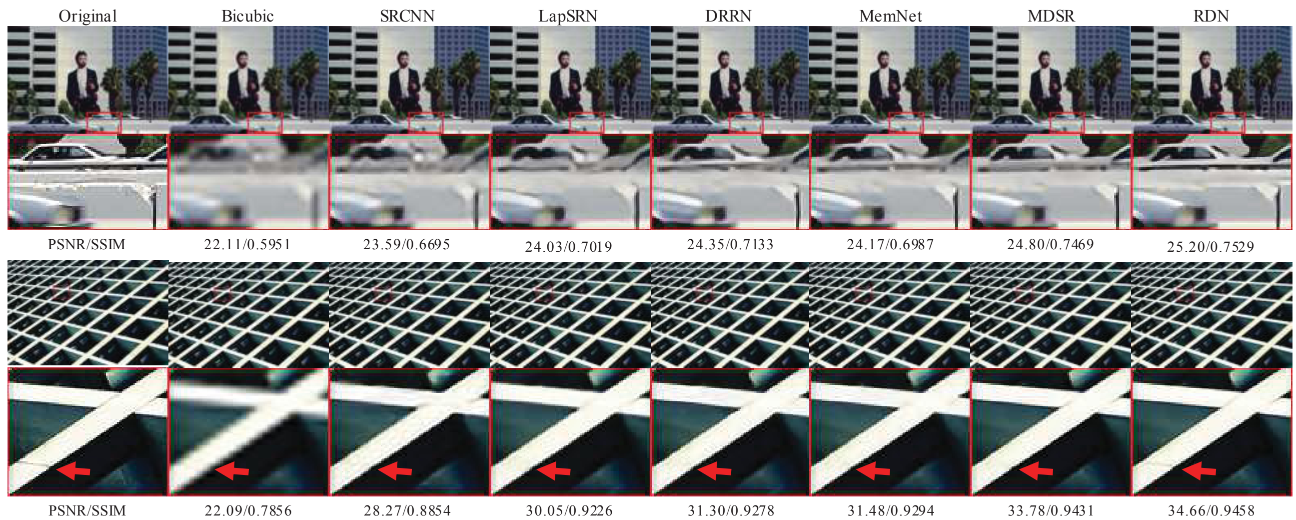


Figure 6. Visual results with **BI** model (×4). The SR results are for image "119082" from B100 and "img_043" from Urban100 respectively.

the effectiveness of our RDN and fair comparison, we don't use deeper network, multi-scale information, or larger input patch size. Moreover, our RDN+ can achieve further improvement with self-ensemble [17].

In Fig. 6, we show visual comparisons on scale ×4. For image "119082", we observe that most of compared methods would produce noticeable artifacts and produce blurred edges. In contrast, our RDN can recover sharper and clearer edges, more faithful to the ground truth. For the tiny line (pointed by the red arrow) in image "'img_043', all the compared methods fail to recover it. While, our RDN can recover it obviously. This is mainly because RDN uses hierarchical features through dense feature fusion.

5.5. Results with BD and DN Degradation Models

Following [38], we also show the SR results with BD degradation model and further introduce DN degradation model. Our RDN is compared with SPMSR [20], SRCNN [3], FSRCNN [4], VDSR [10], IRCNN_G [38], and IRCNN_C [38]. We re-train SRCNN, FSRCNN, and VDSR for each degradation model. Table 3 shows the average PSNR and SSIM results on Set5, Set14, B100, Urban100, and Manga109 with scaling factor ×3. Our RDN and RDN+ perform the best on all the datasets with BD and DN degradation models. The performance gains over other state-of-the-art methods are consistent with the visual results in Figs. 7 and 8.

For **BD** degradation model (Fig. 7), the methods using interpolated LR image as input would produce noticeable artifacts and be unable to remove the blurring artifacts. In contrast, our RDN suppresses the blurring artifacts and recovers sharper edges. This comparison indicates that extracting hierarchical features from the original LR image would alleviate the blurring artifacts. It also demonstrates the strong ability of RDN for **BD** degradation model.

For **DN** degradation model (Fig. 8), where the LR image is corrupted by noise and loses some details. We observe that the noised details are hard to recovered by other methods [3, 10, 38]. However, our RDN can not only handle the noise efficiently, but also recover more details. This comparison indicates that RDN is applicable for jointly image denoising and SR. These results with **BD** and **DN** degradation models demonstrate the effectiveness and robustness of our RDN model.

Dataset	Model	Bicubic	SPMSR [20]	SRCNN [3]	FSRCNN [4]	VDSR [10]	IRCNN_G [38]	IRCNN_C [38]	RDN (ours)	RDN+ (ours)
Set5	**BD**	28.78/0.8308	32.21/0.9001	32.05/0.8944	26.23/0.8124	33.25/0.9150	33.38/0.9182	33.17/0.9157	34.58/0.9280	**34.70/0.9289**
	DN	24.01/0.5369	-/-	25.01/0.6950	24.18/0.6932	25.20/0.7183	25.70/0.7379	27.48/0.7925	28.47/0.8151	**28.55/0.8173**
Set14	**BD**	26.38/0.7271	28.89/0.8105	28.80/0.8074	24.44/0.7106	29.46/0.8244	29.63/0.8281	29.55/0.8271	30.53/0.8447	**30.64/0.8463**
	DN	22.87/0.4724	-/-	23.78/0.5898	23.02/0.5856	24.00/0.6112	24.45/0.6305	25.92/0.6932	26.60/0.7101	**26.67/0.7117**
B100	**BD**	26.33/0.6918	28.13/0.7740	28.13/0.7736	24.86/0.6832	28.57/0.7893	28.65/0.7922	28.49/0.7886	29.23/0.8079	**29.30/0.8093**
	DN	22.92/0.4449	-/-	23.76/0.5538	23.41/0.5556	24.00/0.5749	24.28/0.5900	25.55/0.6481	25.93/0.6573	**25.97/0.6587**
Urban100	**BD**	23.52/0.6862	25.84/0.7856	25.70/0.7770	22.04/0.6745	26.61/0.8136	26.77/0.8154	26.47/0.8081	28.46/0.8582	**28.67/0.8612**
	DN	21.63/0.4687	-/-	21.90/0.5737	21.15/0.5682	22.22/0.6096	22.90/0.6429	23.93/0.6950	24.92/0.7364	**25.05/0.7399**
Manga109	**BD**	25.46/0.8149	29.64/0.9003	29.47/0.8924	23.04/0.7927	31.06/0.9234	31.15/0.9245	31.13/0.9236	33.97/0.9465	**34.34/0.9483**
	DN	23.01/0.5381	-/-	23.75/0.7148	22.39/0.7111	24.20/0.7525	24.88/0.7765	26.07/0.8253	28.00/0.8591	**28.18/0.8621**

Table 3. Benchmark results with **BD** and **DN** degradation models. Average PSNR/SSIM values for scaling factor ×3.

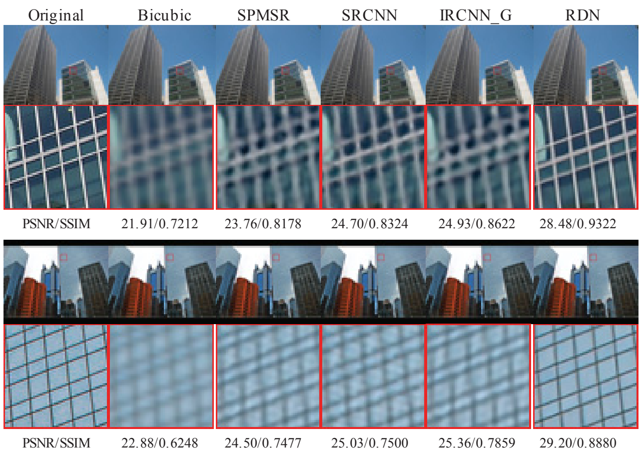


Figure 7. Visual results using **BD** degradation model with scaling factor ×3. The SR results are for image "img_096" from Urban100 and "img_099" from Urban100 respectively.

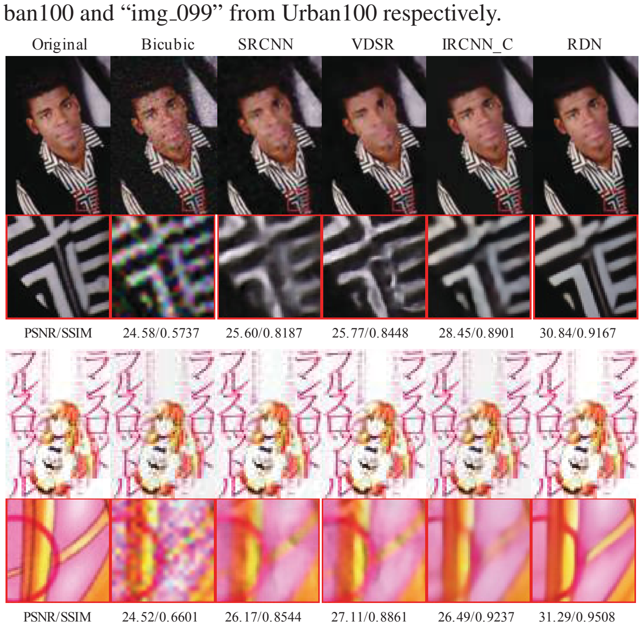


Figure 8. Visual results using **DN** degradation model with scaling factor ×3. The SR results are for image "302008" from B100 and "LancelotFullThrottle" from Manga109 respectively.

5.6. Super-Resolving Real-World Images

We also conduct SR experiments on two representative real-world images, "chip" (with 244×200 pixels) and "hatc" (with 133×174 pixels) [41]. In this case, the original HR images are not available and the degradation model is unknown either. We compare our RND with VDSR [10], LapSRN [13], and MemNet [26]. As shown in Fig. 9, our RDN recovers sharper edges and finer details than other state-of-the-art methods. These results further indicate the benefits of learning dense features from the original input image. The hierarchical features perform robustly for different or unknown degradation models.

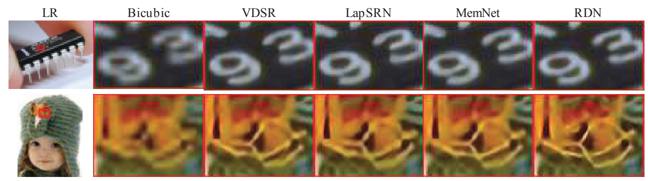


Figure 9. Visual results on real-world images with scaling factor ×4. The two rows show SR results for images "chip" and "hatc" respectively.

6. Conclusions

In this paper, we proposed a very deep residual dense network (RDN) for image SR, where residual dense block (RDB) serves as the basic build module. In each RDB, the dense connections between each layers allow full usage of local layers. The local feature fusion (LFF) not only stabilizes the training wider network, but also adaptively controls the preservation of information from current and preceding RDBs. RDB further allows direct connections between the preceding RDB and each layer of current block, leading to a contiguous memory (CM) mechanism. The local residual leaning (LRL) further improves the flow of information and gradient. Moreover, we propose global feature fusion (GFF) to extract hierarchical features in the LR space. By fully using local and global features, our RDN leads to a dense feature fusion and deep supervision. We use the same RDN structure to handle three degradation models and real-world data. Extensive benchmark evaluations well demonstrate that our RDN achieves superiority over state-of-the-art methods.

7. Acknowledgements

This research is supported in part by the NSF IIS award 1651902, ONR Young Investigator Award N00014-14-1-0484, and U.S. Army Research Office Award W911NF-17-1-0367.

References

[1] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012.

[2] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014.

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 2016.

[4] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016.

[5] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[7] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

[8] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015.

[9] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *submitted to ICLR 2018*, 2017.

[10] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[11] J. Kim, J. Kwon Lee, and K. Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016.

[12] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[13] 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.

[14] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[15] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[16] K. Li, Z. Wu, K.-C. Peng, J. Ernst, and Y. Fu. Tell me where to look: Guided attention inference network. *arXiv preprint arXiv:1802.10171*, 2018.

[17] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017.

[18] D. Martin, 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*, 2001.

[19] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 2017.

[20] T. Peleg and M. Elad. A statistical prediction model based on sparse representations for single image super-resolution. *TIP*, 2014.

[21] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *CVPR*, 2015.

[22] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.

[23] W. Shi, J. Caballero, C. Ledig, X. Zhuang, W. Bai, K. Bhatia, A. M. S. M. de Marvao, T. Dawes, D. ORegan, and D. Rueckert. Cardiac image super-resolution with global correspondence using multi-atlas patchmatch. In *MICCAI*, 2013.

[24] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[25] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017.

[26] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, 2017.

[27] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, 2017.

[28] R. Timofte, V. De, and L. V. Gool. Anchored neighborhood regression for fast example-based super-resolution. In *ICCV*, 2013.

[29] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014.

[30] R. Timofte, R. Rothe, and L. Van Gool. Seven ways to improve example-based single image super resolution. In *CVPR*, 2016.

[31] T. Tong, G. Li, X. Liu, and Q. Gao. Image super-resolution using dense skip connections. In *ICCV*, 2017.

[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004.

[33] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Proc. 7th Int. Conf. Curves Surf.*, 2010.

[34] H. Zhang and V. M. Patel. Densely connected pyramid dehazing network. In *CVPR*, 2018.

[35] H. Zhang and V. M. Patel. Density-aware single image deraining using a multi-stream dense network. In *CVPR*, 2018.

[36] H. Zhang, V. Sindagi, and V. M. Patel. Image de-raining using a conditional generative adversarial network. *arXiv preprint arXiv:1701.05957*, 2017.

[37] K. Zhang, X. Gao, D. Tao, and X. Li. Single image super-resolution with non-local means and steering kernel regression. *TIP*, 2012.

[38] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep cnn denoiser prior for image restoration. In *CVPR*, 2017.

[39] K. Zhang, W. Zuo, and L. Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR*, 2018.

[40] L. Zhang and X. Wu. An edge-guided image interpolation algorithm via directional filtering and data fusion. *TIP*, 2006.

[41] Y. Zhang, Y. Zhang, J. Zhang, D. Xu, Y. Fu, Y. Wang, X. Ji, and Q. Dai. Collaborative representation cascade for single image super-resolution. *IEEE Trans. Syst., Man, Cybern., Syst.*, PP(99):1–11, 2017.

[42] W. W. Zou and P. C. Yuen. Very low resolution face recognition problem. *TIP*, 2012.