This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

Residual Dense Network for Image Restoration

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu, *Fellow, IEEE*

Abstract—Recently, deep convolutional neural network (CNN) has achieved great success for image restoration (IR) and provided hierarchical features at the same time. However, most deep CNN based IR models do not make full use of the hierarchical features from the original low-quality images, thereby resulting in relatively-low performance. In this work, we propose a novel and efficient residual dense network (RDN) to address this problem in IR, by making a better tradeoff between efficiency and effectiveness in exploiting the hierarchical features from all the convolutional layers. Specifically, we propose residual dense block (RDB) to extract abundant local features via densely 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 mechanism. To adaptively learn more effective features from preceding and current local features and stabilize the training of wider network, we proposed local feature fusion in RDB. After fully obtaining dense local features, we use global feature fusion to jointly and adaptively learn global hierarchical features in a holistic way. We demonstrate the effectiveness of RDN with several representative IR applications, single image super-resolution, Gaussian image denoising, image compression artifact reduction, and image deblurring. Experiments on benchmark and real-world datasets show that our RDN achieves favorable performance against state-of-the-art methods for each IR task quantitatively and visually.

Index Terms—Residual dense network, hierarchical features, image restoration, image super-resolution, image denoising, compression artifact reduction, image deblurring.

1 Introduction

SINGLE image restoration (IR) aims to generate a visually pleasing high-quality (HQ) image from its degraded low-quality (LQ) measurement (*e.g.*, downsaled, noisy, compressed, or/and blurred images). Image restoration plays an important and fundamental role and has been widely used in computer vision, ranging from security and surveillance imaging [1], medical imaging [2], to image generation [3]. However, IR is very challenging, because the image degradation process is irreversible, resulting in an ill-posed inverse procedure. To tackle this problem, lots of works have been proposed, such as model-based [4], [5], [6], [7] and learning-based [8], [9] methods. Recently, deep convolutional neural network (CNN) has been widely investigated and achieves promising performance in various image restoration tasks, such as image super-resolution (SR) [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], image denoising (DN) [8], [9], [21], [22], [23], [24], image compression reduction (CAR) [21], [23], [25], and image deblurring [5], [6], [7], [8].

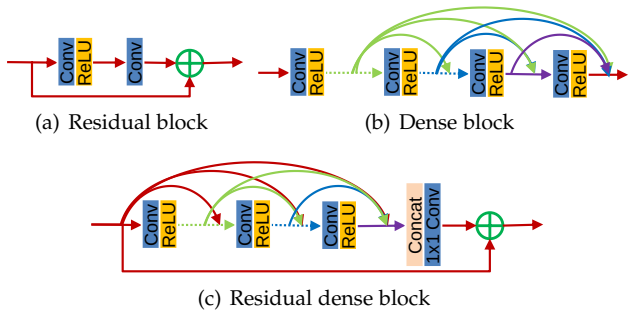
The first challenge is how to introduce deep CNN for image restoration. Dong et al. [15], for the first time, proposed SRCNN for image SR with three convolutional (Conv) layers and achieved significant improvement over previous methods. After firstly introducing CNN for image SR, Dong et al. [25] further applied CNN for other image restoration tasks, like image CAR. But, it's hard to train by stacking more Conv layers in SRCNN. To ease the difficulty of training deep network, Kim et al. proposed VDSR [18] and DRCN [26] by using gradient clipping, residual learning, or recursive-supervision. With newly investigated effective building modules and training strategies, better image SR performance could be further achieved. Incorporating residual learning, Zhang et al. [23] proposed efficient denoising network. Memory block was proposed by Tai et al. to build MemNet for image restoration [9]. Later, Lim et al. proposed simplified residual block (Figure 1(a) by removing batch normalization layers and came up with a very deep network MDSR [27]. Lim et al. further investigated a very wide network EDSR [27] by using residual scaling [28] to stabilizing the training. Convolutional layers in different depth have different size of receptive fields, resulting in hierarchical features. However, these features are not fully used by previous methods. Although MemNet [9] involved a gate unit in memory block to control short-term memory, it failed to build direct connections between local Conv layers and their subsequent ones. As a result, memory block didn't make full use of the features from all the Conv layers within the block either.

What's more, same or similar objects would appear differently in the images due to different scales, angles of view, and aspect ratios. Hierarchical features can capture such characteristics, which would contribute to better reconstruction. However, most learning-based methods (*e.g.*, VDSR [18], LapSRN [29], IRCNN [8], and EDSR [27]) neglect to pursue more clues by making better use of hierarchical features. Although MemNet [9] takes features from several memory block, it failed to extract multi-level features from the original LQ image (*e.g.*, the LR image). Let's take image SR as an example, MemNet would pre-process the original input by interpolating it to desired size. Such a step would not only introduce extra artifacts (*e.g.*, blurring artifacts),

- *Y. Zhang is with Department of ECE, Northeastern University, Boston, MA 02115. E-mail: yulun100@gmail.com*
- *Y. Tian is with Department of Computer Science, University of Rochester, Rochester, NY 14627. E-Mail: yapengtian@rochester.edu*
- *Y. Kong is with the B. Thomas Golisano College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, NY 14623. E-Mail: yu.kong@rit.edu*
- *B. Zhong is with School of Computer Science and Technology, Huaqiao University, Xiamen 361021, China. E-Mail: bnzhong@hqu.edu.cn*
- *Y. Fu is with Department of ECE and College of CIS, Northeastern University, Boston, MA 02115. E-Mail: yunfu@ece.neu.edu*

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) Residual block (b) Dense block

(c) Residual dense block

Fig. 1. Comparison of prior network structures (a,b) and our residual dense block (c). (a) Residual block in MDSR [27]. (b) Dense block in SRDenseNet [19]. (c) Our proposed residual dense block (RDB), which not only enables previous RDB to connect with each layer of current RDB, but also makes better use of local features.

but also increase the computation complexity quadratically. Later, dense block (Figure 1(b)) was introduced in SRDenseNet for image SR [19]. The growth rate is a key part in dense block. Based on the investigation in DenseNet [30] and our experiments (see Section 5.1), higher growth rate contributes to better performance, resulting in wider networks. However, training a wider network with dense blocks would become harder. As investigated in EDSR [27], increasing feature map number above a certain level would make the training more difficult and numerically unstable.

To tackle the issues and limitations above, we propose a simple yet efficient residual dense network (RDN) (Figure 2) by extracting the hierarchical features from the original LQ image. Our RDN is built on our proposed residual dense block (Figure 1(c)). A straightforward way to obtain hierarchical feature is to directly extract the output of each Conv layer in the LQ space. But, it's impractical, especially for a very deep network. Instead, we propose residual dense block (RDB), which consists of dense connected layers and local feature fusion (LFF) with local residual learning (LRL). Furthermore, the Conv layers of current RDB have direct access to the previous RDB, resulting in a contiguous state pass. We name it as contiguous memory mechanism, which further passes on information that needs to be preserved [30]. So, in each RDB, LFF concatenates the states of preceding and current RDBs and adaptively extracts local dense features. Moreover, LFF helps to stabilize the training of wider networks with high growth rate. After obtaining multi-level local dense features, global feature fusion (GFF) is conducted to adaptively preserve the hierarchical features in a global way. As shown in Figures 2 and 3, the original LR input directly connects each layer, leading to an implicit deep supervision [31].

In summary, our main contributions are three-fold:

- We propose a unified framework residual dense network (RDN) for high-quality image restoration. The network makes full use of all the hierarchical features from the original LQ 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 better 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 LQ image.

A preliminary version of this work has been presented as a conference version [32]. In the current work, we incorporate additional contents in significant ways:

- We investigate a flexible structure of RDN and apply it for different IR tasks. Such IR applications allow us to further investigate the potential breadth of RDN.
- We investigate more details and add considerable analyses to the initial version, such as block connection, network parameter number, and running time.
- We extend RDN for Gaussian image denoising, compression artifact reduction, and image deblurring. Extensive experiments on benchmark and real-world data demonstrate that our RDN still outperforms existing approaches in these IR tasks.

2 Related Work

Existing IR methods mainly include model-based [4], [5], [6], [7] and learning-based [8], [9] methods. Among them, deep learning (DL)-based methods have achieved dramatic advantages against conventional methods in computer vision [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46]. Here, we focus on several representative image restoration tasks, such as image super-resolution (SR), denoising (DN), compression artifact reduction (CAR), and image deblurring.

2.1 Image Super-Resolution

Deep learning was first investigated for image SR. The most challenging part is how to formulize the mapping between LR and HR images into deep neural network. After constructing the basic networks for image SR, there're still challenging problems to solve, such as how to train bery deep and wide network. We will show how these challenges are solved or alleviated.

Dong et al. [15] proposed SRCNN, applying CNN into image SR for the first time. Based on SRCNN, lots of improvements have been down to pursue better performance. By using residual learning to ease the training of deeper networks, VDSR [18] and IRCNN [8] could train deeper networks with more stacked Conv layers. DRCN [26] also achieved such a deep network by sharing network parameters and recursive learning, which was further employed in DRRN [47]. However, these methods would pre-process the original input by interpolating it to desired size. Such a step would not only introduce extra artifacts (*e.g.*, blurring artifacts), but also increase the computation complexity quadratically. As a result, extracting features from the interpolated LR images would not be able to build direct mapping from the original LR to HR images.

To address the problem above, Dong et al. [48] built direct mapping from the original LR image to the HR target by introducing a transposed Conv layer (also known as

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

deconvolution layer). Such a transposed Conv layer was further replaced by an efficient sub-pixel convolution layer in ESPCN [49]. Due to its efficiency, sub-pixel Conv layer was adopted in SRResNet [50], which took advantage of residual learning [51]. By extracting features from the original LR input and upscaling the final LR features with transposed or sub-pixel convolution layer, they could 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 neglect to adequately utilize information from each Conv layer, but only upscale the high-level CNN features for the final reconstruction.

2.2 Deep Convolutional Neural Network (CNN)

LeCun et al. integrated constraints from task domain to enhance network generalization ability for handwritten zip code recognition [52], which can be viewed as the pioneering usage of CNNs. Later, various network structures were proposed with better performance, such as AlexNet [53], VGG [54], and GoogleNet [55]. Recently, He et al. [51] investigated the powerful effectiveness of network depth and proposed deep residual learning for very deep trainable networks. Such a very deep residual network achieves significant improvements on several computer vision tasks, like image classification and object detection. Huang et al. proposed DenseNet, which allows direct connections between any two layers within the same dense block [30]. 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 [9] and dense blocks [19]. More differences between DenseNet/SRDenseNet/MemNet and our RDN would be discussed in Section 4.

2.3 Deep Learning for Image Restoration

Generally, there are two categories of methods for image restoration: model-based [4], [5], [6], [7] and learning-based [8], [9] methods. The model-based methods often formulate the image restoration problems as optimization ones. Numerous regularizers have been investigated to search for better solutions, such as sparsity-based regularizers with learned dictionaries [4] and nonlocal self-similarity inspired ones [6]. Also, instead of using explicitly designed regularizers, Dong et al. [7] proposed a denoising prior driven network for image restoration. Then, we give more details about learning-based methods.

Dong et al. [25] proposed ARCNN for image compression artifact reduction (CAR) with several stacked convolutional layers. Mao et al. [22] proposed residual encoder-decoder networks (RED) with symmetric skip connections, which made the network go deeper (up to 30 layers). Zhang et al. [23] proposed DnCNN to learn mappings from noisy images to noise and further improved performance by utilizing batch normalization [56]. Zhang et al. [8] proposed to learn deep CNN denoiser prior for image restoration (IRCNN) by integrating CNN denoisers into model-based optimization method. However, such methods have limited network depth (*e.g.*, 30 for RED, 20 for DnCNN, and 7 for IRCNN), limiting the network ability. Simply stacking more layers cannot reach better results due to gradient vanishing problem. On the other hand, by using short term and long-term memory, Tai et al. [9] proposed MemNet for image restoration, where the network depth reached 212 but obtained limited improvement over results with 80 layers. For 31×31 input patches from 91 images, training an 80-layer MemNet takes 5 days using 1 Tesla P40 GPU [9].

The aforementioned DL-based image restoration methods have achieved significant improvement over conventional methods, but most of them lose some useful hierarchical features from the original LQ 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 LQ space efficiently.

3 Residual Dense Network for IR

In Figure 2, we show our proposed RDN for image restoration, including image super-resolution (see Figure 2(a)), denoising, compression artifact reduction, and deblurring (see Figure 2(b)).

3.1 Network Structure

we mainly take image SR as an example and give specific illustrations for image DN and CAR cases.

RDN for image SR. As shown in Figure 2(a), our RDN mainly consists of four parts: shallow feature extraction net, residual dense blocks (RDBs), dense feature fusion (DFF), and finally the up-sampling net (UPNet). Let's denote I_{LQ} and I_{HQ} 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 LQ input.

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

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

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

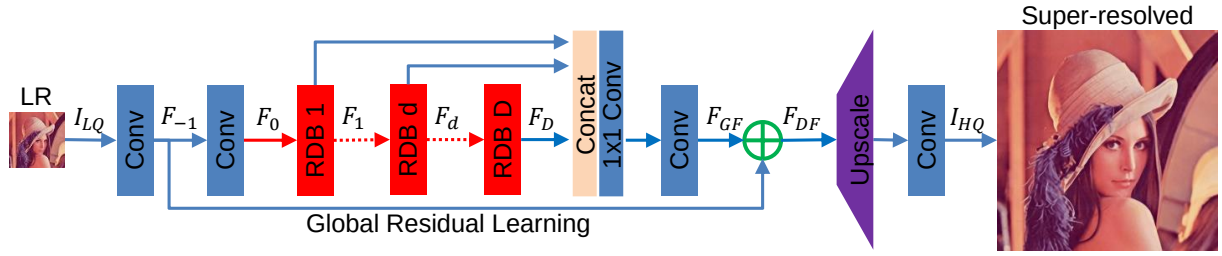
where $H_{RDB,d}$ denotes the operations of the d-th RDB. $H_{RDB,d}$ can be a composite function of operations, such as convolution and rectified linear units (ReLU) [57]. As F_d is produced by the d-th RDB fully utilizing each convolutional layers within the block, we can view F_d as local feature. 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. DFF makes full use of features from all the preceding layers and can be represented as

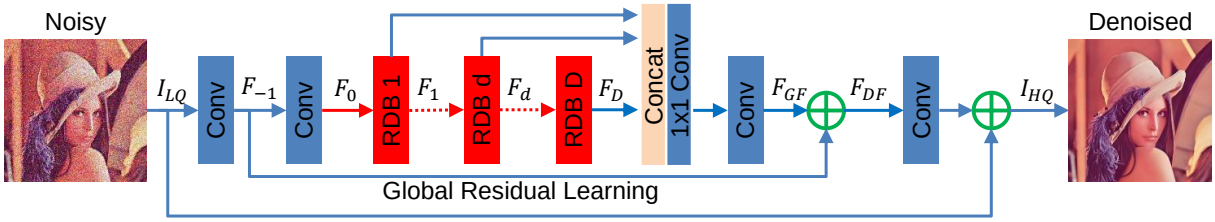
$$F_{DF} = H_{DFF}\left(F_{-1}, F_0, F_1, \cdots, F_D\right), \tag{4}$$

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) RDN for image super-resolution (SR).



(b) RDN for image denoising (DN), compression artifact reduction (CAR), and deblurring. It should be noted that such a structure can also be used for image SR. But, it would consume more resources (*e.g.*, GPU memory and running time). Here, we use image DN as an example.

Fig. 2. The architecture of our proposed residual dense network (RDN) for image restoration.

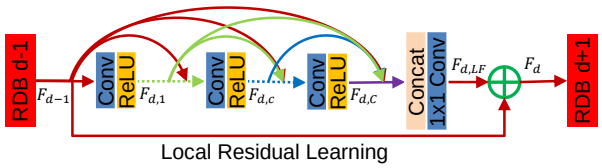


Fig. 3. Residual dense block (RDB) architecture. We denote the connections between the (d-1)-th RDB and the following convolutional layers as contiguous memory (CM) mechanism.

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 LQ space, we stack an up-sampling net (UPNet) in the HQ space. Inspired by [27], we utilize ESPCN [49] in UPNet followed by one Conv layer. The output of RDN can be obtained by

$$I_{HQ} = H_{RDN}\left(I_{LQ}\right), \tag{5}$$

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

RDN for image DN, CAR, and Deblurring. When we apply our RDN to image DN, CAR, and Deblurring, the resolution of the input and output keep the same. As shown in Figure 2(b), we remove the upscaling module in UPNet and obtain the final HQ output via residual learning

$$I_{HQ} = H_{RDN}\left(I_{LQ}\right) + I_{LQ}. \tag{6}$$

Unlike image SR, here, we use residual learning connecting input and output for faster training. It should be noted that the network structure in Figure 2(b) is also suitable for image SR. But, it would consume more GPU memory and running time. As a result, we choose the network structure in Figure 2(a) for image SR.

3.2 Residual Dense Block

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

Contiguous memory (CM) mechanism. The idea behind CM is that we aim at fusing information from all the Conv layers as much as possible. But, directly fuse feature maps from all the Covn layers is not practical, because it would stack huge amount of features. Instead, we turn to first adaptively fuse information locally and then pass them to the following feature fusions. It is realized by passing the state of preceding RDB to each layer of the current RDB. Let F_{d-1} and F_d be the input and output of the d-th RDB respectively and both have G_0 feature-maps. The output of c-th Conv layer of d-th RDB can be formulated as

$$F_{d,c} = \sigma\left(W_{d,c}\left[F_{d-1}, F_{d,1}, \cdots, F_{d,c-1}\right]\right), \tag{7}$$

where σ denotes the ReLU [57] 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 [30]) feature-maps. $\left[F_{d-1}, F_{d,1}, \cdots, F_{d,c-1}\right]$ 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 (LFF). We apply LFF 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 [9], we introduce a 1×1 convolutional layer to adaptively control the output information. We name this operation as local feature fusion (LFF) formulated as

$$F_{d,LF} = H_{LFF}^{d}\left(\left[F_{d-1}, F_{d,1}, \cdots, F_{d,c}, \cdots, F_{d,C}\right]\right), \tag{8}$$

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

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

to train. However, larger growth rate further contributes to the performance, which will be detailed in Section 5.1.

Local residual learning (LRL). We introduce LRL in RDB to further improve the information flow and allow larger growth rate, as there are several convolutional layers in one RDB. The final output of the d-th RDB can be obtained by

$$F_d = F_{d-1} + F_{d,LF}. \tag{9}$$

It should be noted that LRL can also further improve the network representation ability, resulting in better performance. We introduce more results about LRL in Section 5.2. 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 [30] 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. DFF consists of global feature fusion (GFF) and global residual learning.

Global feature fusion (GFF). We propose GFF to extract the global feature F_{GF} by fusing features from all the RDBs

$$F_{GF} = H_{GFF}\left([F_1, \cdots, F_D]\right), \tag{10}$$

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 [50].

Global residual learning. We then utilize global residual learning to obtain the feature-maps before conducting up-scaling by

$$F_{DF} = F_{-1} + F_{GF}, \tag{11}$$

where F_{-1} denotes the shallow feature-maps. All the other layers before global feature fusion are extensively 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 deep dense feature F_{DF}.

It should be noted that Tai et al. [9] utilized long-term dense connections in MemNet to recover more high-frequency information. However, in the memory block [9], the preceding layers don't have direct access to all the subsequent layers. The local feature information is not fully used, limiting the ability of long-term connections. In addition, MemNet extracts features in the HQ space, increasing computational complexity. While, inspired by [27], [29], [48], [49], we extract local and global features in the LQ space. More differences between our proposed RDN and MemNet would be shown in Section 4. We would also demonstrate the effectiveness of global feature fusion in Section 5.2.

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=64 filters and are followed by ReLU [57]. For image SR, following [27], we use ESPCNN [49] to upscale the coarse resolution features to fine ones for the UPNet. For image DN and CAR, the up-scaling module is removed from UPNet. The final Conv layer has 3 output channels, as we output color HQ images. However, the network can also process gray images, for example, when we apply RDN for gray-scale image denoising.

4 Differences with Prior Works

Here, we give more details about the differences between our RDN and several related representative works. We further demonstrate those differences make our RDN more effective in Sections 5 and 6.

Difference to DenseNet. First, DenseNet [30] is widely used in high-level computer vision tasks (*e.g.*, image classification). While RDN is designed more specific for image restoration without using some operations in DenseNet, such as batch normalization and max pooling. Second, DenseNet places transition layers into two adjacent dense blocks. In RDN, the Conv layers are connected by local feature fusion (LFF) and local residual learning. Consequently, the $(d-1)$-th RDB has direct access to each layer in the d-th RDB and also contributes to the input of $(d+1)$-th RDB. Third, we adopt GFF to make better use of hierarchical features, which are neglected in DenseNet.

Difference to SRDenseNet. First, SRDenseNet [19] introduces the basic dense block with batch normalization from DenseNet [30]. Our residual dense block (RDB) improves it in several ways: (1). We propose contiguous memory (CM) mechanism, building direct connections between the preceding RDB and each layer of the current RDB. (2). Our RDB can be easily extended to be wider and be easy to train with usage of local feature fusion (LFF). (3). RDB utilizes local residual learning (LRL) to further encourage the information flow. Second, SRDenseNet densely connects dense blocks, while there are no dense connections among RDBs. We use global feature fusion (GFF) and global residual learning to extract hierarchical features, based on the fully extracted extracted local features by RDBs. Third, SRDenseNet aims to minimize L_2 loss. However, bing more powerful for performance and convergence [27], L_1 loss function is utilized in RDN for each image restoration task.

Difference to MemNet. In addition to the different choices for loss fuction between MemNet [9] and RDN, we mainly summarize additional three differences. First, MemNet has to interpolate the original LR to the desired size, introducing extra blurry artifacts. While, RDN directly extracts features from the original LR input, which helps to reduce computational complexity and contribute to better performance. Second, in MemNet, most Conv layers within one recursive unit have no direct access to their preceding layers or memory block. While, each Conv layer receives

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

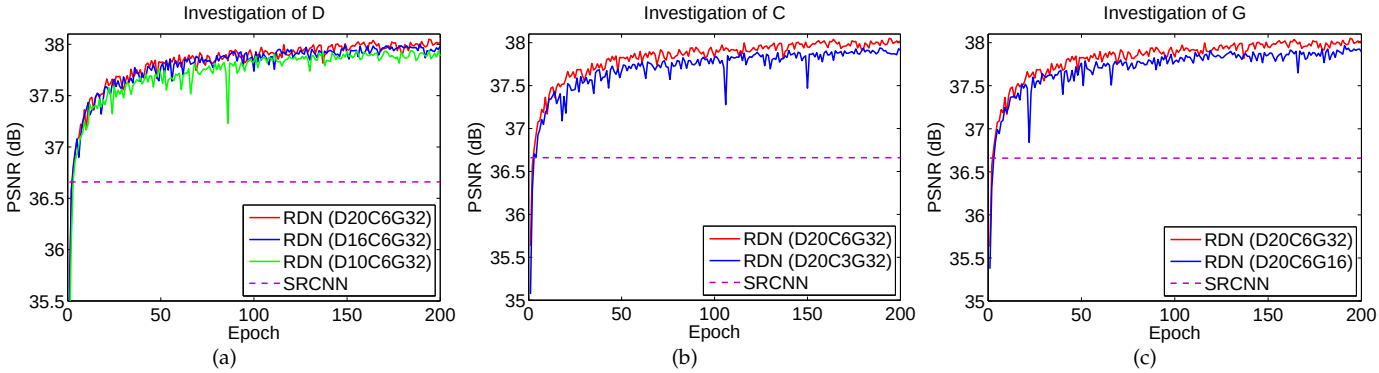


Fig. 4. Convergence analysis of RDN for image SR ($\times 2$) with different values of D, C, and G.

TABLE 1
PSNR (dB) comparisons under different block connections. The results are obtained with Bicubic (**BI**) degradation model for image SR ($\times 4$).

Block connection	Dense connections		Contiguous memory	
Datasets	SRDenseNet [30]	MemNet [25]	RDN (w/o LRL)	RDN (with LRL)
Set5 [58]	32.02	31.74	32.54	32.61
Set14 [59]	28.50	28.26	28.87	28.93
B100 [60]	27.53	27.40	27.75	27.80
Urban100 [17]	26.05	25.50	26.72	26.85

information from preceding RDB and outputs feature for the subsequent layers within same RDB. Plus, LRL further improves the information flow and performance. Third, in MemNet, the memory block neglects to fully utilize the features from preceding block and Conv layers within it. Although dense connections are adopted to connect memory blocks, MemNet fails to extract hierarchical features from the original LR images. However, with the local dense features extracted by RDBs, our RDN further fuses the hierarchical features from the whole preceding layers in a global way in the LR space.

5 Network Investigations

5.1 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 [61] as a reference. As shown in Figures 4(a) and 4(b), larger D or C would lead to higher performance. 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 Figure 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 [61]. More important, our RDN allows deeper and wider network, where more hierarchical features are extracted for higher performance.

5.2 Ablation Investigation

Table 2 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

TABLE 2
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.

	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

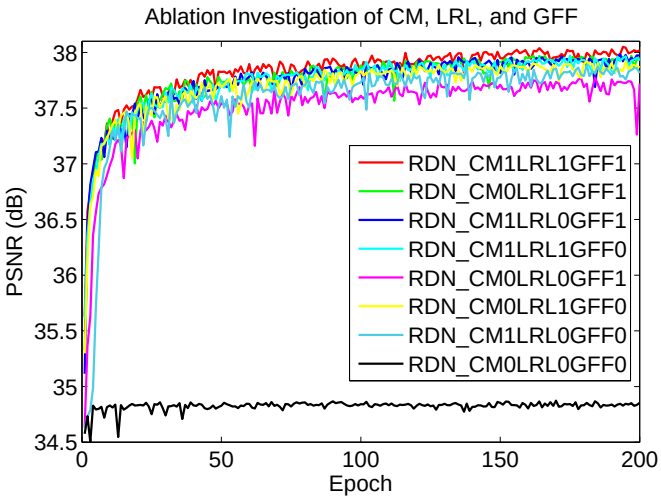


Fig. 5. Convergence analysis on CM, LRL, and GFF. The curves for each combination is based on the PSNR on Set5 ($\times 2$) in 200 epochs.

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 [61] and also demonstrates that stacking many basic dense blocks [30] 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 2nd to 4th combination in Table 2). 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 5th to 7th combination in Table 2). It can be seen that two components

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 3
Parameter number, PSNR, and test time comparisons. The PSNR values are based on Set14 with Bicubic (**BI**) degradation model ($\times 2$).

Methods	LapSRN [29]	DRRN [47]	MemNet [9]	MDSR [27]	EDSR [27]	RDN (ours)
# param.	812K	297K	677K	8M	43M	22M
PSNR (dB)	33.08	33.23	33.28	33.85	33.92	34.14
Time (s)	0.10	17.81	13.84	0.53	1.64	1.56

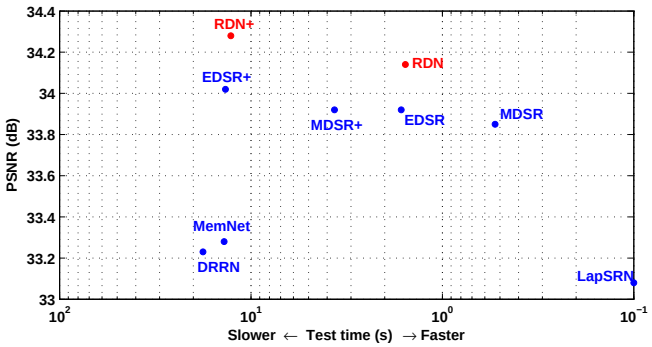


Fig. 6. PSNR and test time on Set14 with **BI** model ($\times 2$).

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 Figure 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.

5.3 Model Size, Performance, and Test Time

We also compare the model size, performance, and test time with other methods on Set14 ($\times 2$) in Table 3. Compared with EDSR, our RDN has half less amount of parameter and obtains better results. Although our RDN has more parameters than that of other methods, RDN achieves comparable (*e.g.*, MDSR) or even less test time (*e.g.*, MemNet). We further visualize the performance and test time comparison in Figure 6. We can see that our RDN achieves good trade-off between the performance and running time.

6 Experimental Results

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

6.1 Settings

Here we provide details of experimental settings, such as training/testing data for each tasks.

Training Data. Recently, Timofte et al. have released a high-quality (2K resolution) dataset DIV2K for image restoration applications [37]. DIV2K consists of 800 training images, 100 validation images, and 100 test images. We use DIV2K as training data for image SR (except for Bicubic degradation model), color and gray image denoising, CAR, and deblurring. For image SR with Bicubic degradation (**BI**), some compared methods (*e.g.*, D-DBPN [38]) further use Flickr2K [27] as well as DIV2K for training. We also train RDN by using larger training data to investigate whether RDN can further improve performance.

Testing Data and Metrics. For testing, we use five standard benchmark datasets: Set5 [58], Set14 [59], B100 [60], Urban100 [17], and Manga109 [64] for image SR. We use Kodak24 (http://r0k.us/graphics/kodak/), BSD68 [60], and Urban100 [17] for color and gray image DN. LIVE1 [65] and Classic5 [66] are used for image CAR. We use McMaster18 [8], Kodak24, and Urban100 as testing data for image deblurring. The quantitative results are evaluated with PSNR and SSIM [67] on Y channel (i.e., luminance) of transformed YCbCr space.

Degradation Models. For image SR, in order to fully demonstrate the effectiveness of our proposed RDN, we use three degradation models to simulate LR images for image SR. 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 $\times 2$, $\times 3$, $\times 4$, and $\times 8$. Similar to [8], 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). For color and gray image denoising, we add Gaussian noise with noise level σ to the ground truth to obtain the noisy inputs. For image CAR, we use Matlab JPEG encoder [68] to generate compressed inputs. For image deburring, the commonly-used 25×25 Gaussian blur kernel of standard deviation 1.6 is used to blur images first. Then, additive Gaussian noise ($\sigma = 2$) is added to the blurry images to obtain the final inputs.

Training Setting. Following settings of [27], in each training batch, we randomly extract 16 LQ RGB patches with the size of 48×48 as inputs for image SR, DN, CAR, and deblurring. 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 [69]. 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.

6.2 Image Super-Resolution

6.2.1 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 state-of-the-art image SR methods: SRCNN [61], FSRCNN [48], SCN [62], VDSR [18], LapSRN [29], MemNet [9], SRDenseNet [19], MSLapSRN [63], EDSR [27], SRMDNF [20], D-DBPN [38], and DPDNN [7]. Similar to [27], [70], we also adopt self-ensemble strategy [27] to further improve our RDN and denote the self-ensembled RDN as RDN+. Here, we also additionally use Flickr2K [37] as training data, which is also used in SRMDNF [20], and D-DBPN [38]. 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

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 4
Quantitative results with BI degradation model. Best and second best results are **highlighted** and underlined.

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	×2	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403	30.80	0.9339
SRCNN [61]	×2	36.66	0.9542	32.45	0.9067	31.36	0.8879	29.50	0.8946	35.60	0.9663
FSRCNN [48]	×2	37.05	0.9560	32.66	0.9090	31.53	0.8920	29.88	0.9020	36.67	0.9710
VDSR [18]	×2	37.53	0.9590	33.05	0.9130	31.90	0.8960	30.77	0.9140	37.22	0.9750
LapSRN [29]	×2	37.52	0.9591	33.08	0.9130	31.08	0.8950	30.41	0.9101	37.27	0.9740
MemNet [9]	×2	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195	37.72	0.9740
DPDNN [7]	×2	37.75	0.9600	33.30	0.9150	32.09	0.8990	31.50	0.9220	N/A	N/A
EDSR [27]	×2	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
SRMDNF [20]	×2	37.79	0.9601	33.32	0.9159	32.05	0.8985	31.33	0.9204	38.07	0.9761
D-DBPN [38]	×2	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
RDN (ours)	×2	38.30	0.9617	34.14	0.9235	32.41	0.9025	33.17	0.9377	39.60	0.9791
RDN+ (ours)	×2	**38.34**	**0.9618**	**34.28**	**0.9241**	**32.46**	**0.9030**	**33.36**	**0.9388**	**39.74**	**0.9794**
Bicubic	×3	30.39	0.8682	27.55	0.7742	27.21	0.7385	24.46	0.7349	26.95	0.8556
SRCNN [61]	×3	32.75	0.9090	29.30	0.8215	28.41	0.7863	26.24	0.7989	30.48	0.9117
FSRCNN [48]	×3	33.18	0.9140	29.37	0.8240	28.53	0.7910	26.43	0.8080	31.10	0.9210
VDSR [18]	×3	33.67	0.9210	29.78	0.8320	28.83	0.7990	27.14	0.8290	32.01	0.9340
LapSRN [29]	×3	33.82	0.9227	29.87	0.8320	28.82	0.7980	27.07	0.8280	32.21	0.9350
MemNet [9]	×3	34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376	32.51	0.9369
DPDNN [7]	×3	33.93	0.9240	30.02	0.8360	29.00	0.8010	27.61	0.8420	N/A	N/A
EDSR [27]	×3	34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653	34.17	0.9476
SRMDNF [20]	×3	34.12	0.9254	30.04	0.8382	28.97	0.8025	27.57	0.8398	33.00	0.9403
RDN (ours)	×3	34.78	0.9299	30.63	0.8477	29.33	0.8107	29.02	0.8695	34.58	0.9502
RDN+ (ours)	×3	**34.84**	**0.9303**	**30.74**	**0.8495**	**29.38**	**0.8115**	**29.18**	**0.8718**	**34.81**	**0.9512**
Bicubic	×4	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577	24.89	0.7866
SRCNN [61]	×4	30.48	0.8628	27.50	0.7513	26.90	0.7101	24.52	0.7221	27.58	0.8555
FSRCNN [48]	×4	30.72	0.8660	27.61	0.7550	26.98	0.7150	24.62	0.7280	27.90	0.8610
VDSR [18]	×4	31.35	0.8830	28.02	0.7680	27.29	0.0726	25.18	0.7540	28.83	0.8870
LapSRN [29]	×4	31.54	0.8850	28.19	0.7720	27.32	0.7270	25.21	0.7560	29.09	0.8900
MemNet [9]	×4	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630	29.42	0.8942
DPDNN [7]	×4	31.72	0.8890	28.28	0.7730	27.44	0.7290	25.53	0.7680	N/A	N/A
SRDenseNet [19]	×4	32.02	0.8930	28.50	0.7780	27.53	0.7337	26.05	0.7819	N/A	N/A
EDSR [27]	×4	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.02	0.9148
SRMDNF [20]	×4	31.96	0.8925	28.35	0.7787	27.49	0.7337	25.68	0.7731	30.09	0.9024
D-DBPN [38]	×4	32.47	0.8980	28.82	0.7860	27.72	0.7400	26.38	0.7946	30.91	0.9137
RDN (ours)	×4	32.61	0.8999	28.93	0.7894	27.80	0.7436	26.85	0.8089	31.45	0.9187
RDN+ (ours)	×4	**32.69**	**0.9007**	**29.01**	**0.7909**	**27.85**	**0.7447**	**27.01**	**0.8120**	**31.74**	**0.9208**
Bicubic	×8	24.40	0.6580	23.10	0.5660	23.67	0.5480	20.74	0.5160	21.47	0.6500
SRCNN [61]	×8	25.33	0.6900	23.76	0.5910	24.13	0.5660	21.29	0.5440	22.46	0.6950
FSRCNN [48]	×8	20.13	0.5520	19.75	0.4820	24.21	0.5680	21.32	0.5380	22.39	0.6730
SCN [62]	×8	25.59	0.7071	24.02	0.6028	24.30	0.5698	21.52	0.5571	22.68	0.6963
VDSR [18]	×8	25.93	0.7240	24.26	0.6140	24.49	0.5830	21.70	0.5710	23.16	0.7250
LapSRN [29]	×8	26.15	0.7380	24.35	0.6200	24.54	0.5860	21.81	0.5810	23.39	0.7350
MemNet [9]	×8	26.16	0.7414	24.38	0.6199	24.58	0.5842	21.89	0.5825	23.56	0.7387
MSLapSRN [63]	×8	26.34	0.7558	24.57	0.6273	24.65	0.5895	22.06	0.5963	23.90	0.7564
EDSR [27]	×8	26.96	0.7762	24.91	0.6420	24.81	0.5985	22.51	0.6221	24.69	0.7841
D-DBPN [38]	×8	27.21	0.7840	25.13	0.6480	24.88	0.6010	22.73	0.6312	25.14	0.7987
RDN (ours)	×8	27.23	0.7854	25.25	0.6505	24.91	0.6032	22.83	0.6374	25.14	0.7994
RDN+ (ours)	×8	**27.40**	**0.7900**	**25.38**	**0.6541**	**25.01**	**0.6057**	**23.04**	**0.6439**	**25.48**	**0.8058**

64 filters per Conv layer, we report results of RDN by using D = 16, C = 8, and G = 64 for a fair comparison.

Table 4 shows quantitative comparisons for ×2, ×3, and ×4 SR. Results of SRDenseNet [19] are cited from their paper. When compared with persistent CNN models (SRDenseNet [19] and MemNet [9]), 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 [19] and the memory block in MemNet [9]. When compared with the remaining models, our RDN also achieves the best average results on most datasets. Specifically, for the scaling factor ×2, our RDN performs the best on all datasets. EDSR [27] uses far more filters (i.e., 256) per Conv layer, leading to a very wide network with a large number of parameters (i.e., 43 M). Our RDN has about half less network parameter number and achieves better performance.

In Figure 7, we show visual comparisons on scales ×4 and ×8. We observe that most of compared methods cannot recover the lost details in the LR image (*e.g.*, "img_004"), even though EDSR and D-DBPN can reconstruct partial details. In contrast, our RDN can recover sharper and clearer edges, more faithful to the ground truth. In image "img_092", some unwanted artifacts are generated in the degradation process. All the compared methods would fail to handle such a case, but enlarge the mistake. However, our RDN can alleviate the degradation artifacts and recover correct structures. When scaling factor goes larger (*e.g.*, ×8), more structural and textural details are lost. Even we human beings can hardly distinguish the semantic content in the LR images. Most compared methods cannot recover the lost details either. However, with the usage of hierarchical features through dense feature fusion, our RDN reconstruct better visual results with clearer structures.

6.2.2 Results with BD and DN Degradation Models

Following [8], we also show the SR results with BD degradation model and further introduce DN degradation model. Our RDN is compared with SPMSR [14], SRCNN [61], FSRCNN [48], VDSR [18], IRCNN_G [8], and IRCNN_C [8]. We re-train SRCNN, FSRCNN, and VDSR for each degradation model. Table 5 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

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

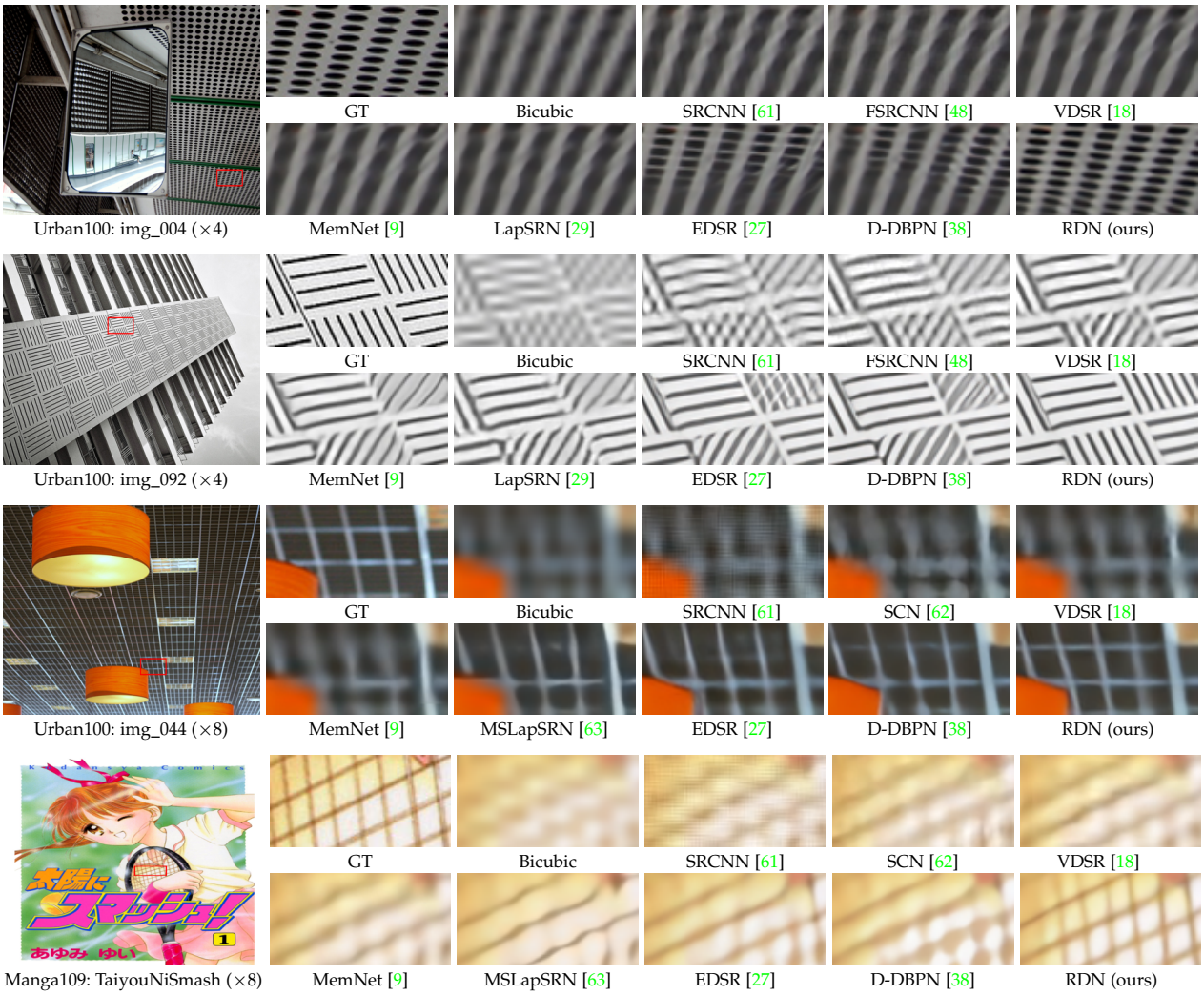


Fig. 7. Image super-resolution results (**BI** degradation model) with scaling factors s = 4 (first two rows) and s = 8 (last two rows).

TABLE 5
Benchmark results with **BD** and **DN** degradation models. Average PSNR/SSIM values for scaling factor $\times 3$.

Dataset	Model	Bicubic	SRCNN [61]	FSRCNN [48]	VDSR [18]	IRCNN_G [8]	IRCNN_C [8]	RDN (ours)	RDN+ (ours)
Set5	**BD**	28.78/0.8308	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	28.80/0.8074	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	28.13/0.7736	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	25.70/0.7770	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	29.47/0.8924	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**

performance gains over other state-of-the-art methods are consistent with the visual results in Figures 8 and 9.

For **BD** degradation model (Figure 8), 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.

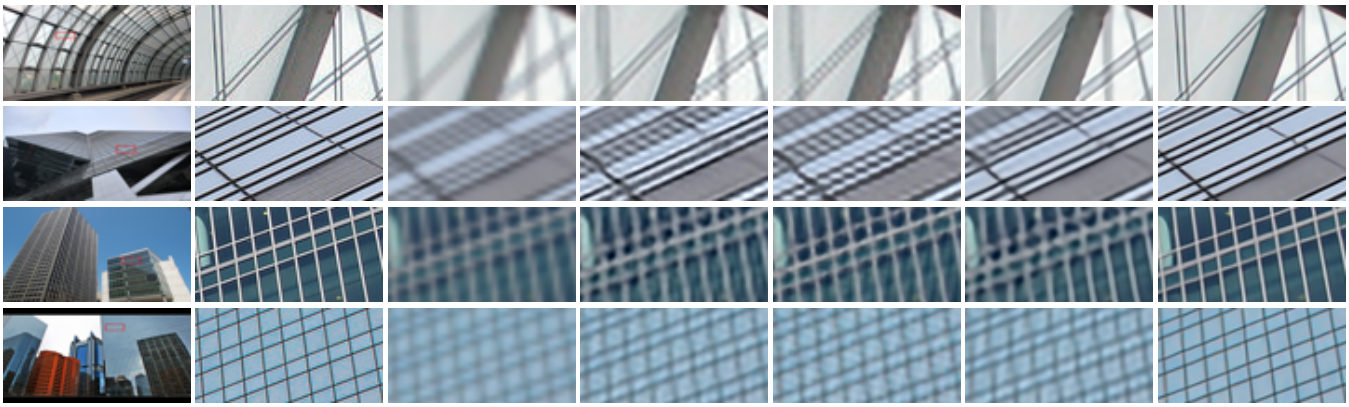
For **DN** degradation model (Figure 9), where the LR image is corrupted by noise and loses some details. We observe that the noised details are hard to recovered by other methods [8], [18], [61]. However, our RDN can not only handle the noise efficiently, but also recover more details. This comparison 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.

6.2.3 *Super-Resolving Real-World Images*

To further demonstrate the effectiveness of our proposed RDN, we super-resolve historical images with JPEG compression artifacts by $\times 4$. We compare with two state-of-the-art methods: SRMDNF [20] and D-DBPN [38]. As shown in Figure 10, the historical image contains letters "HOME" and

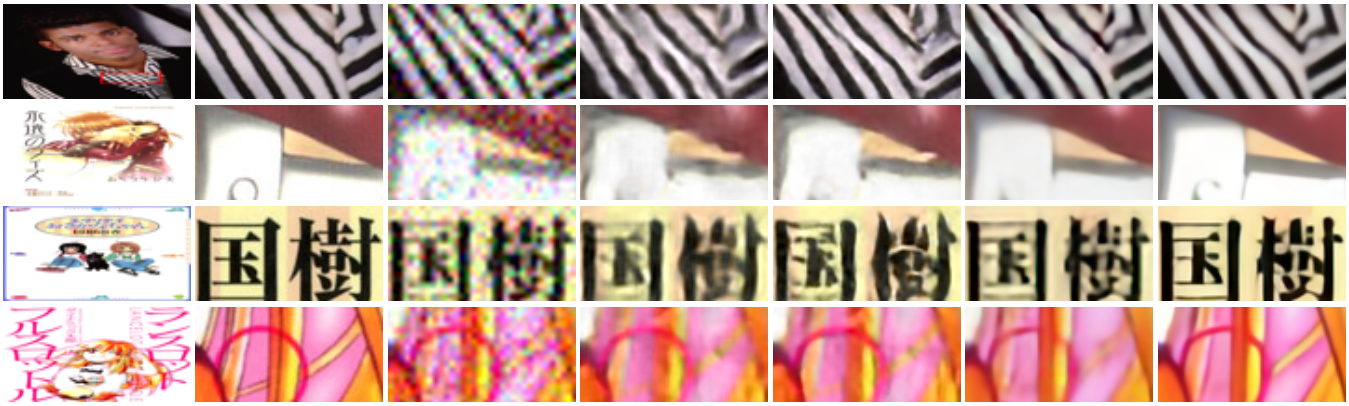
0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence



GT Bicubic SPMSR [14] SRCNN [61] IRCNN [8] RDN (ours)

Fig. 8. Visual results using **BD** degradation model with scaling factor $\times 3$.



GT Bicubic SRCNN [61] VDSR [18] IRCNN [8] RDN (ours)

Fig. 9. Visual results using **DN** degradation model with scaling factor $\times 3$.

Historical: img004 SRMDNF [20] D-DBPN [38] RDN (ours)

Fig. 10. Visual results on real-world images with scaling factor $\times 4$.

"ANTI". Both SRMDNF and D-DBPN suffer from blurring and distorted artifacts. On the other hand, our RDN reconstruct sharper and more accurate results. These comparisons indicate the benefits of learning hierarchical features from the original input, allowing our RDN to perform robustly for different or unknown degradation models.

6.3 Image Denoising

We compare our RDN with recently leading Gaussian denoising methods: BM3D [71], CBM3D [72], TNRD [21], RED [22], DnCNN [23], MemNet [9], IRCNN [8], MWCNN [44], N^3Net [45], NLRN [46], and FFDNet [24]. Kodak24 [1], BSD68 [60], and Urban100 [17] are used for gray-scale and color image denoising. Set12 [23] is also used to test gray-scale image denoising. Noisy images are obtained by adding AWGN noises of different levels to clean images.

6.3.1 Gray-scale Image Denoising

The PSNR results are shown in Table 6. One can see that on all the 4 test sets with 4 noise levels, our RDN+ achieves higher average PSNR values than most of compared methods. On average, for noise level $\sigma = 50$, our RDN achieves 0.28 dB, 0.22 dB, 0.11 dB, and 0.88 dB gains over FFDNet [24] on four test sets respectively. Gains on Urban100 become larger, which is mainly because our method takes advantage of a larger scope of context information with hierarchical features. Moreover, for noise levels σ = 30, 50, and 70, the gains over BM3D of RDN are larger than 0.7 dB, breaking the estimated PSNR bound (0.7 dB) over BM3D in [73]. It should also be noted that our RDN achieves moderate gains over MemNet on BSD68. This is mainly because BSD68 is formed from 100 validation images in Berkeley Segmentation Dataset (BSD) [60]. But MemNet [9] used these 100 validation images and 200 training images in BSD as training data. So it's reasonable for MemNet to perform pretty well on BSD68. Moreover, MWCNN [44], N^3Net [45], and NLRN [46] also have achieved good performance for some noise levels, which indicates that Wavelet transformation and non-local processing are promising candidates for further image denoising improvements.

We show visual gray-scale denoised results of different methods in Figure 11. We can see that BM3D preserves image structure to some degree, but fails to remove noise deeply. TNRD [21] tends to generate some artifacts in the smooth region. RED [22], DnCNN [23], MemNet [9], and IRCNN [8] would over-smooth edges. The main reason should be the limited network ability for high noise level (*e.g.*, σ = 50). In contrast, our RDN can remove noise greatly and recover more details (*e.g.*, the tiny lines in "img_061"). Also, the gray-scale visual results by our RDN in the smooth

1. http://r0k.us/graphics/kodak/

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

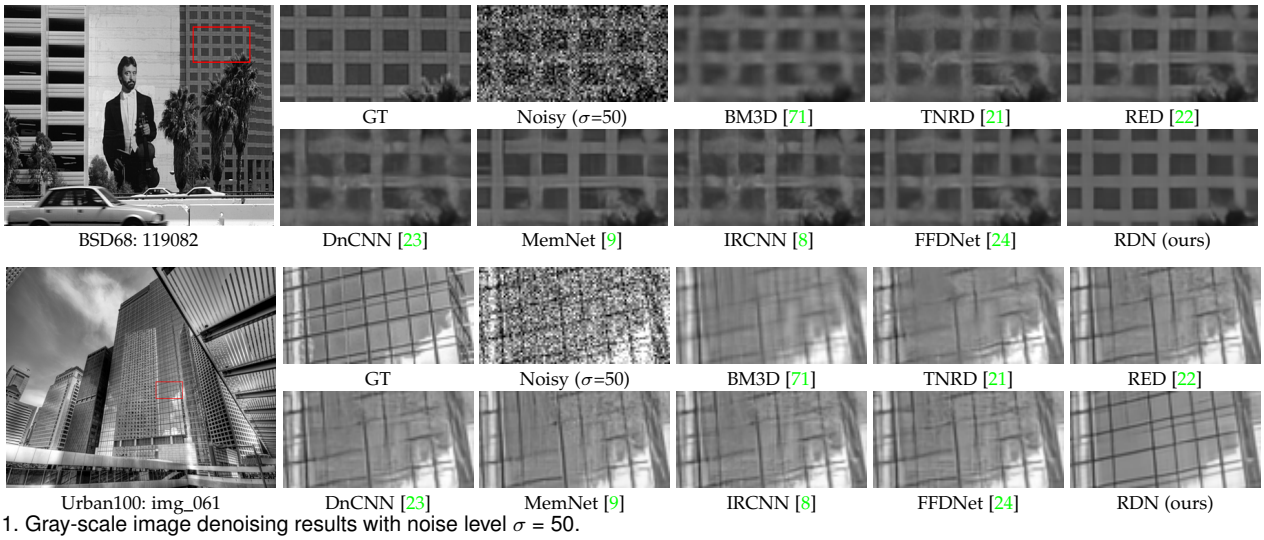


Fig. 11. Gray-scale image denoising results with noise level σ = 50.

region are more faithful to the clean images (*e.g.*, smooth regions in "119082" and "img_061").

6.3.2 Color Image Denoising

We generate noisy color images by adding AWGN noise to clean RGB images with different noise levels σ = 10, 30, 50, and 70. The PSNR results are listed in Table 7. We apply gray image denoising methods (*e.g.*, MemNet [9]) for color image denoising channel by channel. Lager gains over MemNet [9] of our RDN indicate that denoising color images jointly perform better than denoising each channel separately. Take σ=50 as an example, our RDN obtains 0.56 dB, 0.35, and 1.24 dB improvements over FFDNet [24] on three test sets respectively. Residual learning and dense feature fusion allows RDN to go wider and deeper, obtain hierarchical features, and achieve better performance.

We also show color image denoising visual results in Figure 12. CBM3D [72] tends to produce artifacts along the edges. TNRD [21] produces artifacts in the smooth area and is unable to recover clear edges. RED [22], DnCNN [23], MemNet [9], IRCNN [8], and FFDNet [24] could produce blurring artifacts along edges (*e.g.*, the structural lines in "img_039"). Because RED [22] and MemNet [9] were designed for gray image denoising. In our experiments on color image denoising, we conduct RED [22] and MemNet [9] in each channel. Although DnCNN [23], IRCNN [8], and FFDNet [24] directly denoise noisy color images in three channels, they either fail to recover sharp edges and clean smooth area. In contrast, our RDN can recover shaper edges and cleaner smooth area.

6.3.3 Real-World Color Image Denoising

To further demonstrate the effectiveness of our proposed RDN, we compare it with recent leading method MCWNNM [74] on real noisy image. Following the same settings as [74], we estimate the noise levels $(\sigma_r, \sigma_g, \sigma_b)$ via some noise estimation methods [75]. Inspired by [74], we finetune RDN with the noise level σ=$\sqrt{\left(\sigma_r^2+\sigma_g^2+\sigma_b^2\right)/3}$. Due to limited space, we only show the denoised results on the real noisy image "Dog" (with 192×192 pixels) [76], which doesn't have ground truth.

We show real-world color image denoising in Fig. 14. Although MCWNNM considers the specific noise levels of each channel, its result still suffers from over-smoothing artifacts and loses some details (*e.g.*, the moustache). In contrast, our RDN handles the noise better and preserves more details and shaper edges, which indicates that RDN is also suitable for real applications.

Fig. 14. Visual denoised results of the real noisy image "Dog". The noise level of R, G, B channels are estimated as 16.8, 17.0, and 16.6 respectively.

6.4 Image Compression Artifact Reduction

We further apply our RDN to reduce image compression artifacts. We compare our RDN with SA-DCT [66], AR-CNN [25], TNRD [21], and DnCNN [23]. We use Matlab JPEG encoder [68] to generate compressed test images from LIVE1 [65] and Classic5 [66]. Four JPEG quality settings q = 10, 20, 30, 40 are used in Matlab JPEG encoder. Here, we only focus on the compression artifact reduction (CAR) of Y channel (in YCbCr space) to keep fair comparison with other methods.

We report PSNR/SSIM values in Table 8. As we can see, our RDN and RDN+ achieve higher PSNR and SSIM values on LIVE1 and Classic5 with all JPEG qualities than other compared methods. Taking q = 10 as an example, our RDN achieves 0.48 dB and 0.60 dB improvements over DnCNN [23] in terms of PSNR. Even in such a challenging case (very low compression quality), our RDN can still obtain great performance gains over others. Similar improvements are also significant for other compression qualities. These comparisons further demonstrate the effectiveness of our proposed RDN.

Visual comparisons are further shown in Figure 13, where we provide comparisons under very low image quality (q=10). Although ARCNN [25], TNRD [21], and DnCNN [23] can remove blocking artifacts to some degree, they also over-smooth some details (*e.g.*, 1st and 2nd rows

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 6
Quantitative results about gray-scale image denoising. Best and second best results are **highlighted** and underlined

Method	Set12				Kodak24				BSD68				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70	10	30	50	70
BM3D [71]	34.38	29.13	26.72	25.22	34.39	29.13	26.99	25.73	33.31	27.76	25.62	24.44	34.47	28.75	25.94	24.27
TNRD [21]	34.27	28.63	26.81	24.12	34.41	28.87	27.20	24.95	33.41	27.66	25.97	23.83	33.78	27.49	25.59	22.67
RED [22]	34.89	29.70	27.33	25.80	35.02	29.77	27.66	26.39	33.99	28.50	26.37	25.10	34.91	29.18	26.51	24.82
DnCNN [23]	34.78	29.53	27.18	25.52	34.90	29.62	27.51	26.08	33.88	28.36	26.23	24.90	34.73	28.88	26.28	24.36
MemNet [9]	N/A	29.63	27.38	25.90	N/A	29.72	27.68	26.42	N/A	28.43	26.35	25.09	N/A	29.10	26.65	25.01
IRCNN [8]	34.72	29.45	27.14	N/A	34.76	29.53	27.45	N/A	33.74	28.26	26.15	N/A	34.60	28.85	26.24	N/A
MWCNN [44]	N/A	N/A	**27.74**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	**26.53**	N/A	N/A	N/A	27.42	N/A
N^3Net [45]	N/A	N/A	27.43	25.90	N/A	N/A	N/A	N/A	N/A	N/A	26.39	**25.14**	N/A	N/A	26.82	25.15
NLRN [46]	N/A	N/A	27.64	N/A	N/A	N/A	N/A	N/A	N/A	N/A	26.47	N/A	N/A	29.94	27.38	25.66
FFDNet [24]	34.65	29.61	27.32	25.81	34.81	29.70	27.63	26.34	33.76	28.39	26.30	25.04	34.45	29.03	26.52	24.86
RDN (ours)	35.06	29.94	27.60	26.05	35.17	30.00	27.85	26.54	34.00	28.56	26.41	25.10	35.41	30.01	27.40	25.64
RDN+ (ours)	**35.08**	**29.97**	27.64	**26.09**	**35.19**	**30.02**	**27.88**	**26.57**	**34.01**	**28.58**	26.43	25.12	**35.45**	**30.08**	**27.47**	**25.71**

TABLE 7
Quantitative results about color image denoising. Best and second best results are **highlighted** and underlined

Method	Kodak24				BSD68				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70
CBM3D [72]	36.57	30.89	28.63	27.27	35.91	29.73	27.38	26.00	36.00	30.36	27.94	26.31
TNRD [21]	34.33	28.83	27.17	24.94	33.36	27.64	25.96	23.83	33.60	27.40	25.52	22.63
RED [22]	34.91	29.71	27.62	26.36	33.89	28.46	26.35	25.09	34.59	29.02	26.40	24.74
DnCNN [23]	36.98	31.39	29.16	27.64	36.31	30.40	28.01	26.56	36.21	30.28	28.16	26.17
MemNet [9]	N/A	29.67	27.65	26.40	N/A	28.39	26.33	25.08	N/A	28.93	26.53	24.93
IRCNN [8]	36.70	31.24	28.93	N/A	36.06	30.22	27.86	N/A	35.81	30.28	27.69	N/A
FFDNet [24]	36.81	31.39	29.10	27.68	36.14	30.31	27.96	26.53	35.77	30.53	28.05	26.39
RDN (ours)	37.31	31.94	29.66	28.20	36.47	30.67	28.31	26.85	36.69	31.69	29.29	27.63
RDN+ (ours)	**37.33**	**31.98**	**29.70**	**28.24**	**36.49**	**30.70**	**28.34**	**26.88**	**36.75**	**31.78**	**29.38**	**27.74**

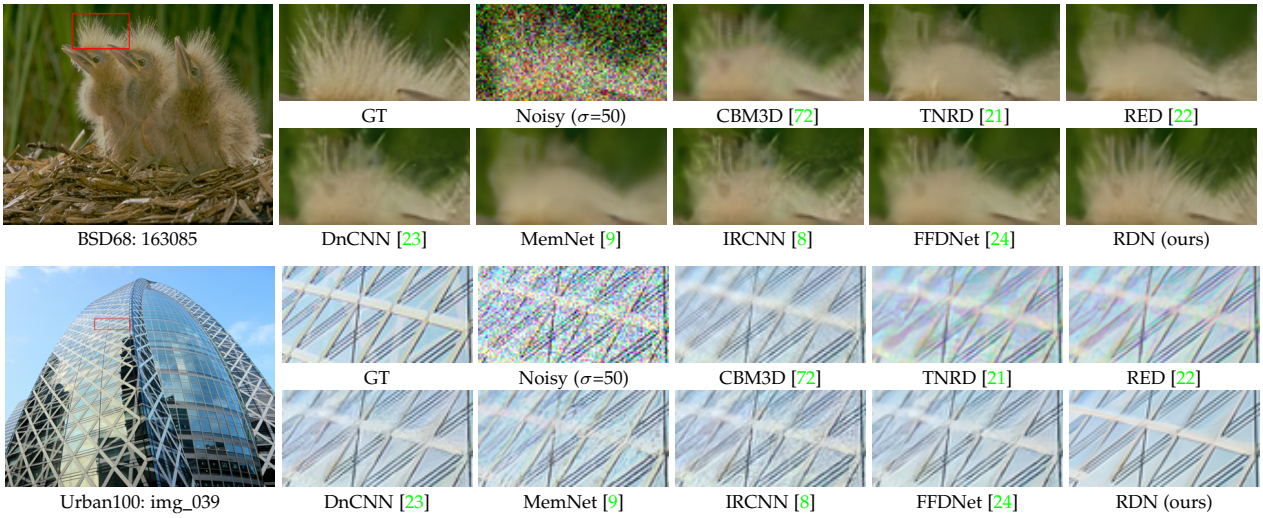


Fig. 12. Color image denoising results with noise level σ = 50.

TABLE 8
Quantitative results about image compression artifact reduction. Best and second best results are highlighted and underlined

Dataset	Quality	JPEG		SA-DCT [66]		ARCNN [25]		TNRD [21]		DnCNN [23]		RDN (ours)		RDN+ (ours)	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
LIVE1	10	27.77	0.7905	28.65	0.8093	28.98	0.8217	29.15	0.8111	29.19	0.8123	29.67	0.8247	**29.70**	**0.8252**
	20	30.07	0.8683	30.81	0.8781	31.29	0.8871	31.46	0.8769	31.59	0.8802	32.07	0.8882	**32.10**	**0.8886**
	30	31.41	0.9000	32.08	0.9078	32.69	0.9166	32.84	0.9059	32.98	0.9090	33.51	0.9153	**33.54**	**0.9156**
	40	32.35	0.9173	32.99	0.9240	33.63	0.9306	N/A	N/A	33.96	0.9247	34.51	0.9302	**34.54**	**0.9304**
Classic5	10	27.82	0.7800	28.88	0.8071	29.04	0.8111	29.28	0.7992	29.40	0.8026	30.00	0.8188	**30.03**	**0.8194**
	20	30.12	0.8541	30.92	0.8663	31.16	0.8694	31.47	0.8576	31.63	0.8610	32.15	0.8699	**32.19**	**0.8704**
	30	31.48	0.8844	32.14	0.8914	32.52	0.8967	32.78	0.8837	32.91	0.8861	33.43	0.8930	**33.46**	**0.8932**
	40	32.43	0.9011	33.00	0.9055	33.34	0.9101	N/A	N/A	33.77	0.9003	34.27	0.9061	**34.29**	**0.9063**

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

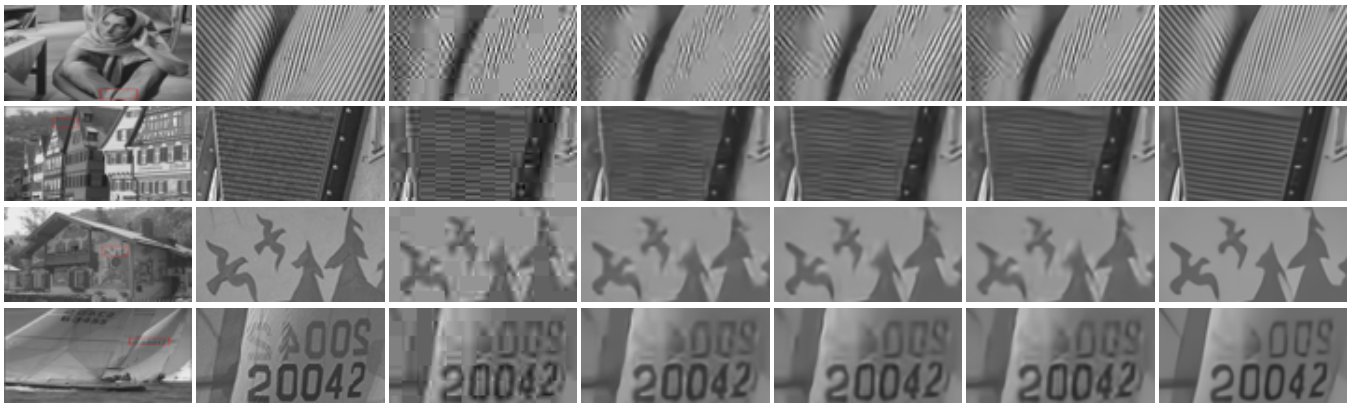


Fig. 13. Image compression artifacts reduction results with JPEG quality q = 10.

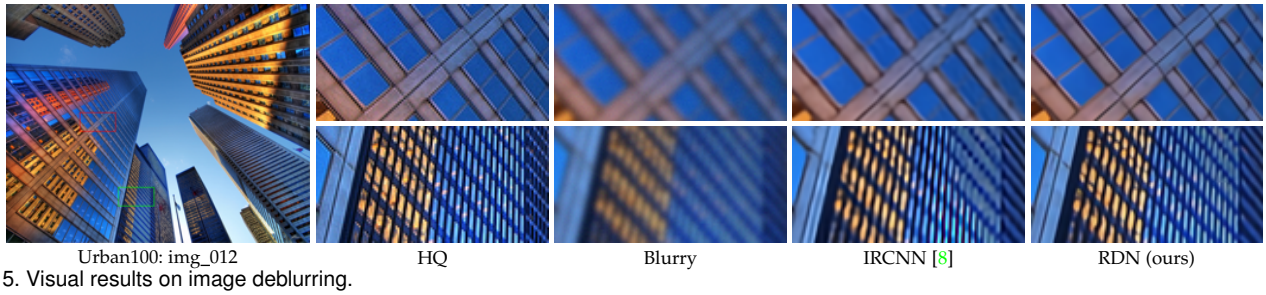


Fig. 15. Visual results on image deblurring.

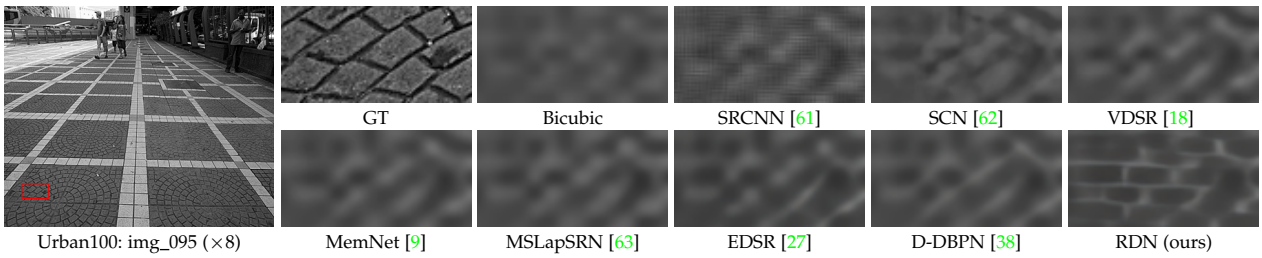


Fig. 16. Failure cases for image super-resolution (×8).

TABLE 9
PSNR (dB)/SSIM results about image deblurring.

Set	McMaster18		Kodak24		Urban100	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Blurry	27.00	0.7817	26.09	0.7142	22.38	0.6732
IRCNN [8]	32.50	0.8961	30.40	0.8513	27.70	0.8577
RDN (ours)	33.60	0.9157	30.88	0.8718	29.34	0.8886

in Figure 13) and cannot deeply remove the compression artifacts around content structures (*e.g.*, 3rd and 4th rows in Figure 13). While, RDN has stronger network representation ability to distinguish compression artifacts and content information better. As a result, RDN recovers more details with consistent content structures.

6.5 Image Deblurring

A common practice to synthesize blurry images is first apply a blur kernel and then add Gausian noise with noise level σ [7], [8]. There are several blur kernels, such as Gaussian and motion blur kernels. Here, we focus on the commonly-used 25×25 Gaussian blur kernel of standard deviation 1.6. The additive Gaussian noise ($\sigma = 2$) is added to the blurry images.

We compare with IRCNN [8] and use McMaster18 [8], Kodak24, and Urban100 as test sets. We provide quantitative results in Table 9, where our RDN achieves large improvements over IRCNN for each test set. We further show visual results in Figure 15, where IRCNN still outputs some blurry structures. While, our RDN reconstructs much sharper edges and tiny details. These quantitative and qualitative comparisons demonstrate that our RDN also performs well for image deblurring.

7 Discussions

Here, we give a brief view of the benefits and limitations of our RDN and challenges in image restoration.

Benefits of RDN. RDB serves as the basic build module in RDN, which takes advantage of local and global feature fusion, obtaining very powerful representational ability. RDN uses less network parameters than residual network while achieves better performance than dense network, leading to a good tradeoff between model size and performance. RDN can be directly applied or generilized to several image restoration tasks with promissing performance.

Limitations of RDN. In some challenging cases (*e.g.*, large scaling factor), RDN may fail to obtain proper details. As shown in Figure 16, although other methods fail to recover proper structures, our RDN cannot generates right structures either. The main reasons of this failure case may be that our RDN cannot recover similar textures based on

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

the limited input information. Instead, RDN would generate most likely texture patterns learned from the training data.

Challenges in Image Restoration. Extreme cases make the image restoration tasks much harder, such as very large scaling factors for image SR, heavy noise for image DN, low JPEG quality in image CAR, and heavy blurring artifacts in image deblurring. Complex desegregation processes in the real world make it difficult for us to formulate the degradation process. Then it may make the data preparation and network training harder.

Future Works. We believe that RDN can be further applied to other image restoration tasks, such as image demosaicing, derain, and dehazing. On the other hand, it's worth to further improve the performance of RDN. As indicated in Figure 16, RDN generates better local structures than others, while the global recovered structures are wrong. How to learn stronger local and global feature representations is worth to investigate. Plus, RDN may further benefit from adversarial training, which may help to alleviate some blurring and over-smoothing artifacts.

8 Conclusions

In this work, based on our proposed residual dense block (RDB), we propose deep residual dense network (RDN) for image restoration. RDB allows direct connections from preceding RDB to each Conv layer of current RDB, which results in a contiguous memory (CM) mechanism. We proposed LFF to adaptively preserve the information from the current and previous RDBs. With the usage of LRL, the flow of gradient and information can be further improved and training wider network becomes more stable. Extensive benchmark and real-world evaluations well demonstrate that our RDN achieves superiority over state-of-the-art methods for several image restoration tasks.

Acknowledgments

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] W. W. Zou and P. C. Yuen, "Very low resolution face recognition problem," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 327–340, Jan. 2012.

[2] 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 *Medical Image Computing and Computer Assisted Intervention*, 2013.

[3] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *Proc. International Conference on Learning Representations*, 2018.

[4] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. on Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[5] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE Trans. on Image Process.*, vol. 22, no. 4, pp. 1620–1630, 2012.

[6] W. Dong, G. Shi, Y. Ma, and X. Li, "Image restoration via simultaneous sparse coding: Where structured sparsity meets gaussian scale mixture," *Int. J. Comput. Vis.*, vol. 114, no. 2-3, pp. 217–232, 2015.

[7] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2305–2318, 2019.

[8] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[9] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017.

[10] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *IEEE Trans. Image Process.*, 2006.

[11] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *IEEE Trans. Image Process.*, 2012.

[12] R. Timofte, V. De, and L. V. Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013.

[13] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. IEEE Asian Conf. Comput. Vis.*, 2014.

[14] T. Peleg and M. Elad, "A statistical prediction model based on sparse representations for single image super-resolution." *IEEE Trans. Image Process.*, 2014.

[15] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014.

[16] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015.

[17] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015.

[18] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[19] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017.

[20] K. Zhang, W. Zuo, and L. Zhang, "Learning a single convolutional super-resolution network for multiple degradations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[21] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[22] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016.

[23] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[24] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn based image denoising," *arXiv preprint arXiv:1710.04026*, 2017.

[25] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 576–584.

[26] J. Kim, J. Kwon Lee, and K. Mu Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[27] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshop*, 2017.

[28] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *Association for the Advancement of Artificial Intelligence*, 2017.

[29] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[30] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[31] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. International Conference on Artificial Intelligence and Statistics*, 2015.

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

[32] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

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

[34] H. Zhang and V. M. Patel, "Density-aware single image de-raining using a multi-stream dense network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[35] ——, "Densely connected pyramid dehazing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[36] 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.

[37] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshop*, 2017.

[38] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[39] C. Ancuti, C. O. Ancuti, R. Timofte, L. Van Gool, L. Zhang, M.-H. Yang, V. M. Patel, H. Zhang, V. A. Sindagi, R. Zhao *et al.*, "Ntire 2018 challenge on image dehazing: Methods and results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshop*, 2018.

[40] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, "2018 pirm challenge on perceptual image super-resolution," in *Proc. Eur. Conf. Comput. Vis. Workshop*, 2018.

[41] K. Yu, C. Dong, L. Lin, and C. C. Loy, "Crafting a toolchain for image restoration by deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 2443–2452.

[42] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proc. Eur. Conf. Comput. Vis. Workshop*, 2018.

[43] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018.

[44] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-cnn for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshop*, 2018.

[45] T. Plötz and S. Roth, "Neural nearest neighbors networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018.

[46] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018.

[47] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[48] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2016.

[49] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[50] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[52] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 1–9.

[56] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015.

[57] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. International Conference on Artificial Intelligence and Statistics*, 2011.

[58] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.*, 2012.

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

[60] 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 *Proc. IEEE Int. Conf. Comput. Vis.*, 2001.

[61] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2016.

[62] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015.

[63] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PP, no. 99, pp. 1–14, 2018.

[64] 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.

[65] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik, "Live image quality assessment database release 2 (2005)," 2005.

[66] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images," *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1395–1411, 2007.

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

[68] J. Jancsary, S. Nowozin, and C. Rother, "Loss-specific training of non-parametric image restoration models: A new state of the art," in *Proc. Eur. Conf. Comput. Vis.* Springer, Oct. 2012, pp. 112–125.

[69] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations*, 2014.

[70] R. Timofte, R. Rothe, and L. Van Gool, "Seven ways to improve example-based single image super resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[71] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. Image Process.*, 2007.

[72] ——, "Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space," in *Proc. IEEE Int. Conf. Image Process.*, 2007.

[73] A. Levin, B. Nadler, F. Durand, and W. T. Freeman, "Patch complexity, finite pixel correlations and optimal denoising," in *Proc. Eur. Conf. Comput. Vis.*, 2012.

[74] J. Xu, L. Zhang, D. Zhang, and X. Feng, "Multi-channel weighted nuclear norm minimization for real color image denoising," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017.

[75] G. Chen, F. Zhu, and P. Ann Heng, "An efficient statistical method for image noise level estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015.

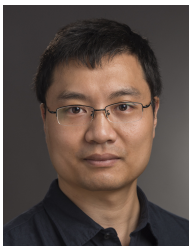
[76] M. Lebrun, M. Colom, and J.-M. Morel, "The noise clinic: a blind image denoising algorithm," *Image Processing On Line*, vol. 5, pp. 1–54, 2015.

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.2968521, IEEE Transactions on Pattern Analysis and Machine Intelligence

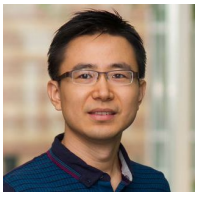
Yulun Zhang received B.E. degree from School of Electronic Engineering, Xidian University, China, in 2013 and M.E. degree from Department of Automation, Tsinghua University, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of ECE, Northeastern University, USA. He was the receipt of the Best Student Paper Award at IEEE International Conference on Visual Communication and Image Processing(VCIP) in 2015. He also won the Best Paper Award at IEEE International Conference on Computer Vision (ICCV) RLQ Workshop in 2019. His research interests include image restoration and deep learning.

Yapeng Tian received the B.E. degree in electronic engineering from Xidian University, Xian, China, in 2013, M.E. degree in electronic engineering at Tsinghua University, Beijing, China, in 2017, and is currently working toward the PhD degree in department of computer science at University of Rochester, USA. His research interests include audio-visual scene understanding and low-level vision.



Yu Kong received B.Eng. degree in automation from Anhui University in 2006, and PhD degree in computer science from Beijing Institute of Technology, China, in 2012. He is now a tenure-track Assistant Professor in the B. Thomas Golisano College of Computing and Information Sciences at Rochester Institute of Technology. Prior to that, he visited the National Laboratory of Pattern Recognition (NLPR), Chinese Academy of Science, and the Department of Computer Science and Engineering, University at Buffalo, SUNY. He was a postdoc in the Department of ECE, Northeastern University. Dr. Kong's research interests include computer vision, social media analytics, and machine learning. He is a member of the IEEE.

Bineng Zhong received the B.S., M.S., and Ph.D. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 2004, 2006, and 2010, respectively. From 2007 to 2008, he was a Research Fellow with the Institute of Automation and Institute of Computing Technology, Chinese Academy of Science. From September 2017 to September 2018, he was a visiting scholar in Northeastern University, Boston, MA, USA. Currently, he is an professor with the School of Computer Science and Technology, Huaqiao University, Xiamen, China. His research interests include pattern recognition, machine learning, and computer vision.



Yun Fu (S'07-M'08-SM'11-F'19) received the B.Eng. degree in information engineering and the M.Eng. degree in pattern recognition and intelligence systems from Xian Jiaotong University, China, respectively, and the M.S. degree in statistics and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, respectively. He is an interdisciplinary faculty member affiliated with College of Engineering and the College of Computer and Information Science at Northeastern University since 2012. His research interests are Machine Learning, Computational Intelligence, Big Data Mining, Computer Vision, Pattern Recognition, and Cyber-Physical Systems. He has extensive publications in leading journals, books/book chapters and international conferences/workshops. He serves as associate editor, chairs, PC member and reviewer of many top journals and international conferences/workshops. He received seven Prestigious Young Investigator Awards from NAE, ONR, ARO, IEEE, INNS, UIUC, Grainger Foundation; nine Best Paper Awards from IEEE, IAPR, SPIE, SIAM; many major Industrial Research Awards from Google, Samsung, and Adobe, etc. He is currently an Associate Editor of the IEEE Transactions on Neural Networks and Leaning Systems (TNNLS). He is fellow of IEEE, IAPR, OSA and SPIE, a Lifetime Distinguished Member of ACM, Lifetime Member of AAAI and Institute of Mathematical Statistics, member of ACM Future of Computing Academy, Global Young Academy, AAAS, INNS and Beckman Graduate Fellow during 2007-2008.

0162-8828 (c) 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.