Multi-scale Residual Hierarchical Dense Networks for Single Image Super-Resolution

CHUANGCHUANG LIU1, XIANFANG SUN2, CHANGYOU CHEN3, PAUL L. ROSIN2, YITONG YAN1, LONGCUN JIN1 (Member, IEEE), XINYI PENG1
1School of Software Engineering, South China University of Technology, Guangzhou, China
2School of Computer Science and Informatics Cardiff University, UK
3Department of Computer Science and Engineering, University at Buffalo, State University of New York, NY, USA
Corresponding author: Longcun Jin (e-mail: lcjin@scut.edu.cn).

ABSTRACT Single image super-resolution is known to be an ill-posed problem, which has been studied for decades. With the developments of deep convolutional neural networks, the CNN-based single image super-resolution methods have greatly improved the quality of the generated high-resolution images. However, it is difficult for image super-resolution to make full use of the relationship between pixels in low-resolution images. To address this issue, we propose a novel multi-scale residual hierarchical dense network, which tries to find the dependencies in multi-level and multi-scale features. Specially, we apply the atrous spatial pyramid pooling, which concatenates multiple atrous convolutions with different dilation rates, and design a residual hierarchical dense structure for single image super-resolution. The atrous-spatial-pyramid-pooling module is used for learning the relationship of features at multiple scales; while the residual hierarchical dense structure, which consists of several hierarchical dense blocks with skip connections, aims to adaptively detect key information from multi-level features. Meanwhile, dense features from different groups are connected in a dense approach by hierarchical dense blocks, which can adequately extract local multi-level features. Extensive experiments on benchmark datasets illustrate the superiority of our proposed method compared with state-of-the-art methods. The super-resolution results on benchmark datasets of our method can be downloaded from https://github.com/Rainyfish/MS-RHDN, and the source code will be released upon acceptance of the paper.

INDEX TERMS Convolutional neural networks, deep learning, multi-scale residual hierarchical dense, image super-resolution

I. INTRODUCTION

SINGLE image super-resolution (SISR) aims to reconstruct a high-resolution (HR) image from its low-resolution (LR) version. Image super-resolution is widely used in many computer vision fields, such as video surveillance, remote sensing, and image sensing. However, SISR is a typically ill-posed problem as the image degradation process is usually irreversible and lots of tiny textures are missing in LR images. Several high-resolution images can be potentially generated from a given LR image. Recently, deep convolutional neural networks have been applied in many tasks, ranging from low-level (image restoration, SISR, etc.) to high-level (image classification, object detection, etc.) vision fields, and have shown great improvements compared with conventional methods.

Currently, CNN-based SISR methods, which learn an effective nonlinear mapping function from LR images to HR images directly, have greatly improved the quality of the super-resolved image. Among them, Dong et al. [1] firstly used a deep convolutional neural network called SRCNN, consisting of three convolutional layers, to address the SISR problem. Since then, lots of deep-learning SISR methods have been developed. VDSR [2] provides remarkable performance by increasing the depth of the network to 20, proving the importance of the network depth for detecting effective features of images. SRCNN and VDSR involve the interpolated images for pre-processing, whose spatial size is the same as the HR images. FSRCNN [3] was proposed to

All these methods try to make full use of image information or features to improve performance, which include increasing the network depth, widening channels, or applying recursive learning. However, most CNN-based SISR models do not take full advantage of the multi-level information from different convolutional layers. Furthermore, these methods usually neglect to use the information from different scales. Objects in images may be similar at different scales and information from different scales may give some clues to help generate high-quality HR images.

To address this issue, we propose a novel network based on the multi-scale structure and residual hierarchical dense connection. The dense connection extracts more information from different layers. We use two levels of dense connections to detect local and global multi-level features. To extract multi-scale features, we simply apply the residual atrous-spatial-pyramid-pooling structure to fully make use of the information from multiple scales in the LR images. For stabilizing the network and easing the training difficult, we use residual learning to detect more informative features.

Overall, the main contributions of our method are three-fold:

- We propose a unified framework multi-scale residual hierarchical dense network for image super-resolution. Our method aims at making full use of multi-scale and multi-level features in the LR input image.
- We propose a residual hierarchical dense module to focus on global and local multi-level features. We use sub-dense blocks (SDBs) to adaptively obtain the essential parts of the dense features. Skip connections are applied for efficient network training and performance improvements.
- We propose a residual multi-scale structure to detect multi-scale features, which can be readily applied to other super-resolution networks. Such multi-scale structure further improves the performance of the network. In addition, our model obtains much better SR performance than previous CNN-based methods.

The remainder of this paper is organized as follows: Section 2 introduces related work on image super-resolution. Section 3 presents the proposed method. Section 4 gives experimental results on benchmark datasets. Visual comparisons with other methods are also included. To show the effectiveness of the components in our network, Section 5 gives the network investigations. Finally, Section 6 draws
II. RELATED WORK

The SISR methods can mainly be categorized into three classes: interpolation-based methods [10]–[13], reconstruction-based methods [14]–[16], and learning-based methods. The interpolation-based methods, such as bilinear interpolation and bicubic interpolation, are simple and fast, but suffer from over-smoothed textures and thus are not able to produce high-quality images. The reconstruction-based methods are flexible and usually use prior knowledge to produce high-frequency details. However, these methods are usually time-consuming and suffer from rapid degeneration of performance with the increasing upsampling factor.

The learning-based methods attempt to learn mappings from LR space to HR space directly. Freeman et al. [17] firstly used Markov random fields (MRF) to generate synthetic images, where the parameters of the model are learned from the examples. Chang et al. [18] used locally linear embedding (LLE) [19] to find the resolutions from the linear combination of nearest neighbors. ANR [20] proposed by Timofte et al. uses sparse learned dictionaries and applies the coefficients calculated from LR patches to the corresponding SR patches directly. A+ [21], an improved version of ANR, learns regressors on all training patches. There also exist SR methods based on decision trees or random forests such as [22]–[26] to address the SISR problem.

Recently, deep-learning based methods have shown great improvements in image super-resolution. Specifically, Dong et al. [1] firstly proposed a deep convolutional neural network SRCNN for the SISR problem. The depth of the network plays an important role in many vision tasks, Kim et al. proposed VDSR [2] with remarkable performance, which increased the depth of the network. To reduce the parameters and find the dependencies of different proceeding time, DRRN [9] uses recursive learning and the memory block with the deeper network. Instead of interpolating the original LR images to the desired size before putting them into the networks, FSRCNN [3] extracted features from the original LR images and used a deconvolutional layer to upscale the spatial size at the end of the network, which greatly reduced the computations. This manner is commonly used in recent SR methods. LapSRN [4] progressively reconstructs the HR image with increasing scales of input images. MS-LapSRN [5], an improved version of LapSRN, uses a multi-scale training strategy to handle the multiple upsampling scales in one single model. Shi et al. [27] proposed ESPCN, which introduces a sub-pixel convolutional layer for efficient upsampling. Lim et al. [7] proposed EDSR and a multi-scale deep MDSR, which removed the unnecessary batch normalization layer from the ResNet [6] architecture. SRMDNF [28] is proposed to handle multiple degradations by concatenating degradation maps and images as the input to the network and adaptively learn to produce high-quality images under different blur kernels of downsampling. ZSSR [29] (zero-shot super-resolution) uses an unsupervised approach to learn the mapping from LR images to HR images, whose training data are generated by downsampling the test data. D-DBPN [30] uses an error-correcting feedback mechanism for SR by iterative up and downsampling. RDN [31] proposed by Zhang et al. uses residual and dense connections and achieves state-of-the-art performance. Zhang et al. [32] proposed RNAN, where local and non-local attention blocks are used to adaptively rescale features with soft attention.

In order to produce photo-realistic SR images, Ledig et al. [33] firstly introduced residual learning and generative adversarial network (GAN) to decrease the distance between the distributions of real images and SR images. However, the images generated by SRGAN still contain noise and artifacts. Wang et al. [34] introduced an enhanced SRGAN, which applied relativistic GAN [35] to the discriminator and adopted residual scaling [36], smaller initialization, and network interpolation, to remove artifacts and won the first place in the 2018 PIRM-SR challenge in region 3. For producing realistic SR images, many loss functions have been proposed. The perceptual loss [37] is proposed to reduce the pixel-wise distances of high-level features produced by pre-trained models, e.g. VGG19 [38]. Contextual loss [39] maintains the image statistics and approximates the KL-divergence.

Making full use of the information in the LR images is the key to produce plausible SR images. To investigate multi-scale and multi-level features, we propose a multi-scale residual hierarchical dense network to obtain results with improvements in quality and quantity. We will introduce our method in the next section.

III. PROPOSED METHOD

In this paper, our method aims to reconstruct a high-resolution image $I_{\text{SR}} \in R^{W_{r} \times H_{r} \times C}$ from a low-resolution image $I_{\text{LR}} \in R^{W \times H \times C}$, where $W$ and $H$ are the width and height of the LR image, $r$ is the upsampling factor, and $C$ is the number of channels of the color space. Fig.1 shows the main framework of our network, whose components are detailed below.
A. NETWORK ARCHITECTURE

Our proposed multi-scale residual hierarchical dense network (MS-RHDN) consists of three main components: shallow feature extraction $F_{SF}$, deep feature extraction $F_{DF}$, and feature reconstruction $F_{REC}$. We use one convolutional layer to extract the shallow features $H_0$, including edges, corners, etc., from $I^{LR}$:

$$H_0 = F_{SF}(I^{LR}), \quad (1)$$

where $H_0$ is the input to the deep feature extraction module. In deep feature extraction, we use $M$ sequential multi-scale residual hierarchical dense blocks (MS-RHDB) and a global fusion layer $F_{GF}$ to extract and fuse multi-scale and multi-level features. Furthermore, a global skip connection is introduced to make the main parts of the network focus on high-frequency information. Formally, we have

$$H_{DF} = F_{DF}(H_0) = H_0 + F_{GF}([H_1, H_2, ..., H_m, ..., H_M]) \quad (2)$$

with $H_m = F_{MSD}^m(H_{m-1})$, where $F_{MSD}^m$ denotes the mapping of the $m$-th MS-RHDB; $[\cdot]$ stands for the concatenation operator. Finally, the feature reconstruction module produces a high-resolution image $I^{SR}$ based on the feature $H_{DF}$:

$$I^{SR} = F_{REC}(H_{DF}) = F_{conv}(F_{up}(H_{DF})). \quad (3)$$

Here the feature reconstruction module is composed of an upsampling layer $F_{up}$ and a convolutional layer $F_{conv}$. There have been a number of advanced upsampling structures, e.g., deconvolutional layer, sub-pixel convolution, EUSR [40]. Here we adopt the sub-pixel convolution, which has been shown effective in previous works such as EDSR [7] and RDN [31].

To define a proper loss function, researchers have designed different loss functions such as $L_2$, $L_1$, perceptual, and adversarial losses. In our work, we choose $L_1$ loss in order to reduce computational complexity. Given a training set $\{I^{LR}_1, I^{HR}_{1,k}\}_{k=1}^N$, where $I^{LR}$ is obtained by down-sampling from $I^{HR}$ with scaling factor $r$, the $L_1$ loss is defined as:

$$L_1(I^{SR}, I^{HR}) = \frac{1}{W_r \times H_r \times C} \times \sum_{c=1}^C \sum_{w=1}^{W_r} \sum_{h=1}^{H_r} ||F_\theta(I^{LR}_{1,k})(w, h, c) - I^{HR}_{1,k}(w, h, c)||_1, \quad (4)$$

where $W_r$, $H_r$, and $C$ stand for the width, height, and channels of the low-resolution image respectively and $r$ is the scaling factor. $F_\theta$ denotes the function of our network and $\theta$ stands for the set of parameters, which are updated by stochastic gradient descent.

B. MULTI-SCALE RESIDUAL HIERARCHICAL DENSE BLOCK (MS-RHDB)

The proposed MS-RHDB framework mainly contains three components: a hierarchical dense module, a memory unit, and a multi-scale block. This section details the hierarchical dense module and the memory unit. Detailed description of the multi-scale block is given in Section III-C.

Hierarchical dense module (HDM) The hierarchical dense module is built to adequately exploit multi-level features. In the module, $K$ sub-dense blocks (SDB) are arranged in a dense manner. Detailed description of SDB is introduced in Section III-D. In general, an HDM takes $H_{m-1}$ as input, and outputs an intermediate feature $H_m^{HDM}$. Formally, this procedure is described as:

$$H_m^{HDM} = F_{HDM}^m(H_{m-1}) = F_{SDB}^K([H_{m-1}, S_1, ..., S_k, ..., S_{K-1}]), \quad (5)$$

where $F_{HDM}^m$ denotes the function of an HDM in the $m$-th MS-RHDB block; and $F_{SDB}^K$ denotes the function of the $K$-th SDB that constitutes $F_{HDM}^m$. $S_k$ denotes the output of the $k$-th SDB, whose input is the concatenation of outputs of the previous $k-1$ SDBs. Formally, $S_k$ can be presented as:

$$S_k = F_{SDB}^K([H_{m-1}, S_1, ..., S_{k-1}]), \quad (6)$$

where $F_{SDB}^K$ denotes the function of the $k$-th SDB. $S_k$ contains $G$ feature-maps, where $G$ is the number of channels and also known as growth rate in [41].

Memory unit. After extracting multi-level features with a set of SDBs, we use a memory unit [42] to integrate these features, which is supposed to adaptively extract unified information. Furthermore, a memory unit is also useful to reduce the number of feature-maps, thus reducing the number of parameters and computations. Specifically, the memory unit is defined as:

$$H_{m}^{MU} = F_{MU}^m([H_{m-1}, S_1, ..., S_k, ..., S_{K}]), \quad (7)$$

where $F_{MU}^m$ denotes the mapping function of the memory unit in the $m$-th MS-RHDB; and $H_m^{MU}$ is the output of $F_{MU}^m$. Following [9] [31], the memory unit is represented with a $1 \times 1$ convolutional layer. Finally, we use a multi-scale structure, which will be introduced in section III-C, to extract...
features from different scales for taking full advantage of fused features by the memory unit. A skip connection is introduced for a similar purpose to the global skip connection. The final output of the $m$-th MS-RHDB is obtained by:

$$H_m = H_{m-1} + F_{m}^{MS}(H_{m}^{MU}),$$

where $F_{MS}^{m}$ denotes the function of the multi-scale block in the $m$-th MS-RHDB; and $H_m^{MS}$ denotes the output of the $m$-th MSB.

### C. MULTI-SCALE BLOCK (MSB)

As discussed above, multi-scale information is useful in generating high-quality super-resolution images. In this section, we elaborate on our multi-scale block used in (8). As shown in Fig.3, the MSB mainly consists of an atrous-spatial-pyramid-pooling (ASPP) structure and a local skip connection.

The ASPP structure is firstly introduced in DeepLabV3 [43] for handling different sizes of objects in street-scene segmentation. ASPP consists of several parallel atrous convolutional layers with different dilated rates. In our model, we apply ASPP to detect useful components of the fused hierarchical dense features. In addition, to make the network efficient and stable, we add a local skip connection to each ASPP. Formally, our multi-scale block is defined as:

$$H_{MS}^{m} = H_{m}^{MU} + F_{ASPP}^{m}(H_{m}^{MU}),$$

where $F_{ASPP}^{m}$ denotes the function of the ASPP structure in the $m$-th MS-RHDB; and $H_m^{MS}$ denotes the output of the $m$-th MSB.

### D. SUB-DENSE BLOCK (SDB)

In order to extract local multi-level features, we introduce a sub-dense neural network. As introduced above, an HDM is constructed by stacking several SDBs in a dense manner, where SDBs are used to extract local multi-level features from previous concatenated features. Because the input channels of each SDB may be different, the number of convolutional layers is determined by the number of the input feature-maps. More feature-maps need more layers. As shown in Fig.4, SDB contains four components: bottleneck-like compression (BLC), local dense group (LDG), input shortcut connection (ISC), and compression shortcut connection (CSC).

Firstly, we use a BLC, which is a bottleneck-like method by a $3 \times 3$ Conv layer for reducing parameters and computations. After compressing the number of feature-maps into $G$, we stack several conv blocks in a manner similar to the DenseNet [41], until the number of feature-maps equals that of the input in LDG. A conv block consists of a concatenation operator applied to all the previous features, a convolutional layer with kernel size of $3 \times 3$, and an activation layer, as shown in Fig.4. The input of the $k$-th SDB, $S_k$, is the concatenation of the outputs of the previous $k-1$ layers and
TABLE I. Quantitative results with the BI degradation model. The best and second best results are highlighted and underlined respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>Scale</th>
<th>Set5</th>
<th>Set14</th>
<th>B100</th>
<th>Urban100</th>
<th>Manga100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PNSR</td>
<td>SSIM</td>
<td>PNSR</td>
<td>SSIM</td>
<td>PNSR</td>
</tr>
<tr>
<td>Bicubic</td>
<td>x2</td>
<td>33.66</td>
<td>0.9299</td>
<td>30.24</td>
<td>0.8088</td>
<td>29.56</td>
</tr>
<tr>
<td>SRCNN [1]</td>
<td>x2</td>
<td>36.66</td>
<td>0.9542</td>
<td>32.45</td>
<td>0.8907</td>
<td>31.90</td>
</tr>
<tr>
<td>FSRCNN [3]</td>
<td>x2</td>
<td>37.05</td>
<td>0.9560</td>
<td>32.66</td>
<td>0.8900</td>
<td>31.53</td>
</tr>
<tr>
<td>VDSR [2]</td>
<td>x2</td>
<td>37.53</td>
<td>0.9590</td>
<td>33.05</td>
<td>0.9130</td>
<td>31.90</td>
</tr>
<tr>
<td>LapSRN [4]</td>
<td>x2</td>
<td>37.52</td>
<td>0.9591</td>
<td>33.08</td>
<td>0.9130</td>
<td>31.08</td>
</tr>
<tr>
<td>MemNet [42]</td>
<td>x2</td>
<td>37.78</td>
<td>0.9597</td>
<td>33.28</td>
<td>0.9142</td>
<td>32.08</td>
</tr>
<tr>
<td>EDSR [7]</td>
<td>x2</td>
<td>38.11</td>
<td>0.9602</td>
<td>33.92</td>
<td>0.9195</td>
<td>32.32</td>
</tr>
<tr>
<td>SRMDNF [28]</td>
<td>x2</td>
<td>37.56</td>
<td>0.9605</td>
<td>33.73</td>
<td>0.9150</td>
<td>32.05</td>
</tr>
<tr>
<td>D-DBPN [30]</td>
<td>x2</td>
<td>38.09</td>
<td>0.9600</td>
<td>33.85</td>
<td>0.9190</td>
<td>32.27</td>
</tr>
<tr>
<td>RDN [31]</td>
<td>x2</td>
<td>38.24</td>
<td>0.9614</td>
<td>34.01</td>
<td>0.9212</td>
<td>32.34</td>
</tr>
<tr>
<td>RNAN [32]</td>
<td>x2</td>
<td>38.17</td>
<td>0.9611</td>
<td>33.87</td>
<td>0.9207</td>
<td>32.32</td>
</tr>
<tr>
<td>SDNND [44]</td>
<td>x2</td>
<td>38.07</td>
<td>0.9610</td>
<td>33.77</td>
<td>0.9195</td>
<td>32.25</td>
</tr>
<tr>
<td>MS-RHDN (ours)</td>
<td>x2</td>
<td>38.26</td>
<td>0.9615</td>
<td>33.92</td>
<td>0.9206</td>
<td>32.36</td>
</tr>
<tr>
<td>MS-RHDN (ours)</td>
<td>x2</td>
<td>38.31</td>
<td>0.9617</td>
<td>34.01</td>
<td>0.9212</td>
<td>32.40</td>
</tr>
<tr>
<td>MS-RHDN+ (ours)</td>
<td>x2</td>
<td>38.31</td>
<td>0.9617</td>
<td>34.01</td>
<td>0.9212</td>
<td>32.40</td>
</tr>
</tbody>
</table>

H_{m-1}, which contains k × G channels. As a result, the LDG in the k-th SDB requires k – 1 convolutional layers to reach the same number of channels as that of the input. Formally, the LDG is described as:

\[ S^L_{D} = F^L_{D}(B_{L}(k-1), S_{k-1,1}, \ldots, S_{k-1,d}, \ldots, S_{k-1,k-1}) \]

where \( B_{L}(k-1) \) is the output of BLC, and \( S_{k-1,d} \) is the output of the d-th Conv in the k-th SDB. \( F^L_{D} \) denotes the function of the local dense group in the k-th SDB and \( S^L_{D} \) is its output. Similarly, the input of the d-th Conv is the concatenation of proceeding layers. In SDB, the local dense group adaptively detects local multi-level features according to the amount of information that the input has.

ISC and CSC. At the same time, the ISC stands for an element-wise addition in the input \( S_{k-1} \) and the multi-level features \( S_{k-1,k-1} \). After that, we apply a 1 × 1 Conv to fuse the number of feature-maps into G. Finally, the compressed feature \( S^G_{k-1} \) is added to the output of the fusion block by CSC. These two shortcut connections are important for detecting more informative cues and improving performance, as well as stabilizing the network.

E. IMPLEMENTATION DETAILS

In this section, we specify some implementation details of our proposed MS-RHDN. We set the kernel size of all the
convolutional layers to $3 \times 3$, except for the fusion layers, whose kernel sizes are set to $1 \times 1$. The number of MS-RHDB is set to $M = 10$. In each MS-RHDB, we set the number of SDBs as $K = 5$. The number of convolutional layers in LDG is decided adaptively, depending on the input. As an illustration, the LDG in the $k$-th SDB stack $k - 1$ convolutional layers organized in a dense manner to get the same number of channels as the input. We set the growth rate as $G = 64$. We use ESPCN [27] to upscale the coarse resolution feature-maps to fine ones in our reconstruction module. At the tail of the network, we use 3 convolutional filters to generate high-quality super-resolved images with 3 color channels.

**Difference to RDN.** Here, we mainly summarize three differences between our method and the RDN [31]. First, both RDN and our model use dense connections. However, multi-level dense connections are adopted in our MS-RHDN, compared to one level in RDN. Second, we apply multi-scale blocks to our MS-RHDN, which aims at extracting multi-scale information from features, while RDN ignores this important information. Third, we proposed SDBs to learn local hierarchical features instead of simple convolutional layers in RDN. Experiments in the next section show that our MS-RHDN outperforms RDN in benchmark datasets with less parameters.

**IV. EXPERIMENTAL RESULTS**

In this section, we conduct quantitative and visual comparisons with several state-of-the-art methods on benchmark datasets under two commonly used image degradations: bicubic downsampling and blur-downsampling, respectively.

**A. SETTINGS**

We use the DIV2K dataset [8] as our training set, which contains 800, 100 and 100 images of 2K-resolution for training, validation, and testing, respectively. The LR images...
are obtained by bicubic downsampling (BI) from the source high-resolution images. At testing, we use five standard benchmark datasets: Set5 [48], Set14 [49], BSD100 [50], Urban100 [51], and Manga109 [52]. We transform the images into YCrCb color space and evaluate the performance by PSNR and SSIM on the Y channel.

In training, images are augmented by rotating and flipping. The batch size is set to 16. Our MS-RHDN is trained based on image patches and optimized with the ADAM optimizer [53]. The hyperparameters $\beta_1$ and $\beta_2$ in the ADAM optimizers are set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We randomly crop $48 \times 48$ patches from LR images as the input of the network. Following [4], [7], [27], [31], [32], the initial learning rate is set to $1 \times 10^{-4}$, which decays to half every $2 \times 10^5$ iterations. We implement our model using the Pytorch [54] framework with a Titan Xp GPU. Training the MS-RHDN roughly takes one day with $2 \times 10^5$ iterations.

### B. COMPARISONS WITH STATE-OF-THE-ART METHODS

We compare our model with 13 state-of-the-art image SR algorithms: SRCNN [1], VDSR [2], FSRCNN [3], SCN [45], LapSRN [4], MemNet [42], EDSDR [7], SRMDNF [28], DBPN [30], RDN [28], RNAN [32], and SDNDF [44]. Similar to [7], [31], [55], we also apply a self-ensemble strategy, which rotates and flips inputs to generate different versions of high-resolution images. The corresponding inverse transforms are applied to generate an HR image, which is an average version of all the HR images. This version is denoted as the self-ensemble MS-RHDN or MS-RHDN+.

#### Quantitative comparison

**Table 2.** Quantitative results with the blur-down degradation model. Best and second best results are highlighted and underlined.

<table>
<thead>
<tr>
<th>Method</th>
<th>Scale</th>
<th>Set5 PSNR</th>
<th>Set5 SSIM</th>
<th>Set14 PSNR</th>
<th>Set14 SSIM</th>
<th>Urban100 PSNR</th>
<th>Urban100 SSIM</th>
<th>Manga109 PSNR</th>
<th>Manga109 SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicubic</td>
<td>$\times 3$</td>
<td>28.78</td>
<td>0.8308</td>
<td>26.38</td>
<td>0.7271</td>
<td>26.33</td>
<td>0.6918</td>
<td>23.52</td>
<td>0.6862</td>
</tr>
<tr>
<td>SPMSR [46]</td>
<td>$\times 3$</td>
<td>32.21</td>
<td>0.9001</td>
<td>28.89</td>
<td>0.8105</td>
<td>28.13</td>
<td>0.7740</td>
<td>25.84</td>
<td>0.7856</td>
</tr>
<tr>
<td>SRCNN [1]</td>
<td>$\times 3$</td>
<td>32.05</td>
<td>0.8944</td>
<td>28.80</td>
<td>0.8074</td>
<td>28.13</td>
<td>0.7736</td>
<td>25.70</td>
<td>0.7770</td>
</tr>
<tr>
<td>FSRCNN [3]</td>
<td>$\times 3$</td>
<td>26.23</td>
<td>0.8124</td>
<td>24.44</td>
<td>0.7106</td>
<td>24.86</td>
<td>0.6832</td>
<td>22.04</td>
<td>0.6745</td>
</tr>
<tr>
<td>VDSR [2]</td>
<td>$\times 3$</td>
<td>33.25</td>
<td>0.9150</td>
<td>29.46</td>
<td>0.8244</td>
<td>28.57</td>
<td>0.7893</td>
<td>26.61</td>
<td>0.8136</td>
</tr>
<tr>
<td>IRCNN [47]</td>
<td>$\times 3$</td>
<td>33.38</td>
<td>0.9182</td>
<td>29.63</td>
<td>0.8281</td>
<td>28.65</td>
<td>0.7922</td>
<td>26.77</td>
<td>0.8154</td>
</tr>
<tr>
<td>RDN [31]</td>
<td>$\times 3$</td>
<td>34.58</td>
<td>0.9280</td>
<td>30.53</td>
<td>0.8447</td>
<td>29.23</td>
<td>0.8079</td>
<td>28.46</td>
<td>0.8582</td>
</tr>
<tr>
<td>MS-RHDN (ours)</td>
<td>$\times 3$</td>
<td>34.76</td>
<td>0.9292</td>
<td>30.67</td>
<td>0.8468</td>
<td>29.32</td>
<td>0.8098</td>
<td>28.83</td>
<td>0.8648</td>
</tr>
<tr>
<td>MS-RHDN+ (ours)</td>
<td>$\times 3$</td>
<td>34.82</td>
<td>0.9298</td>
<td>30.75</td>
<td>0.8481</td>
<td>29.32</td>
<td>0.8108</td>
<td>29.04</td>
<td>0.8679</td>
</tr>
</tbody>
</table>

**Qualitative comparison.** Next, we qualitatively compare our method with state-of-the-art methods. Fig.5 shows the visual comparisons of SR images generated by our method and the methods compared. We obtain several observations from Fig.5. For image ‘img_004’ in Urban100, most compared methods produce images with blurring artifacts. What is worse, most of them cannot recover the detailed textures of the green horizontal line and lattices. However, our method can generate more tiny textures and remove the artifacts. For image ‘img_074’, we can find that most compared methods cannot generate the horizontal line correctly and also suffer from blurring artifacts. Some of them even produce edges with wrong directions. By contrast, our MS-RHDN shows great abilities in producing accurate information from the LR image. For image ‘img_092’, we observe that Bicubic, SRCNN, FSRCNN, VDSR, and LapSRN suffer from blurring artifacts. Even though EDSR, D-DBPN, and SRMDNF can recover some high-frequency information, the right part of the cropped image generated by these methods shows wrong directions of the gap with over-smoothed edges. Our MS-RHDN can be more faithful to the ground truth. For image ‘YumeiroCooking’, due to the abundance of textures, most compared methods cannot fully recover them and obviously produce blurring artifacts. Our method achieves a better result, which is more similar to the HR image.

Overall, our method shows better performance both quantitatively and visually, as it provides a nice way to make full use of features in LR images. Our proposed residual hierarchical dense module successfully detects multi-level features. The multi-scale block is further used to extract information from multiple scales. Multiple residual connections are applied to make the network focus on important parts, and to facilitate the training of the proposed network.

Following [31], [47], we further apply our method to recover images from a blur-down degradation model. A high-resolution image is first blurred by a Gaussian kernel, and then downsampled with a scaling factor. The size of the Gaussian kernel is $7 \times 7$ with standard deviation of 1.6. We compare our method with 6 state-of-the-art methods: SPMSR [46], SRCNN [1], FSRCNN [3], VDSR [2], IRCNN [47], and RDN [31]. Table 2 shows the results of our method and the compared methods in terms of PSNR and SSIM. We observe that our MS-RHDN has much better performance.

In training, images are augmented by rotating and flipping. The batch size is set to 16. Our MS-RHDN is trained based on image patches and optimized with the ADAM optimizer [53]. The hyperparameters $\beta_1$ and $\beta_2$ in the ADAM optimizers are set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We randomly crop $48 \times 48$ patches from LR images as the input of the network. Following [4], [7], [27], [31], [32], the initial learning rate is set to $1 \times 10^{-4}$, which decays to half every $2 \times 10^5$ iterations. We implement our model using the Pytorch [54] framework with a Titan Xp GPU. Training the MS-RHDN roughly takes one day with $2 \times 10^5$ iterations.
performance on all the benchmark datasets, and MS-RHDN+ achieves the best results. Our methods outperform RDN by a large margin. The observations indicate that the structure MS-RHDN is more efficient and has a stronger ability to recover images from the blur-down degradation model. Fig. 6 demonstrates visual comparisons for $3 \times$ SR under the blur-down degradation model. For image ‘img_046’ in Urban100, it is observed that the two patches generated by Bicubic are totally blurred and lose most details. RDN can recover some details but produces some edges with wrong directions. In contrast, MS-RHDN obtains much better performance with sharper edges and correct structures. However, it can be seen from the left parts of the cropped patches on the first line of Fig. 6 that it is challenging for our method and the RDN to recover tiny textures, as these textures are badly blurred in the input LR image. This problem is also the concern in other single image super-resolution methods. Nevertheless, if we zoom in the picture, we can find that our method can still generate edge effect even in the left part of the cropped patch, while the RDN cannot. For image ‘MisutenaideDaisy’ in Manga109, characters in the two patches produced by Bicubic cannot even be recognized by eyeballing. RDN recovers some details of the characters with simple structures, e.g., “c” and “s”. However, some characters in the first patch suffer from blurring artifacts and lose some structures. In comparison, MS-RHDN can recover more details and maintain the right structures. In the second patch, RDN generates characters with over-smoothed edges and blurring artifacts. In contrast, our MS-RHDN alleviates the over-smoothness and blurring-artifacts issues by recovering sharper edges. It also demonstrates the promising potential to make full use of the multi-level and multi-scale features for SR with a blur-down degradation model.

V. NETWORK INVESTIGATIONS

We show ablation investigations on the hierarchical dense module and the multi-scale block in Table 3. The structure 9 has 10 MS-RHDBs ($M=10$), 5 SDBs ($K=5$), and growth rate ($G=32$). We set the batch size to 8 for fast training. To show the superiority of our structure and reduce the influence
of parameters, we design another 8 network structures with approximate parameters by varying the numbers of MS-RHDN and SDB.

Hierarchical dense module (HDM). To demonstrate the effect of our HDM, we add in different configurations with LDG, ISC, or/and CSC. In table 3, when LDG, ISC, and CSC are absent, the PSNR value on Set5 is relatively low. The positive effect of LDG is demonstrated by the performance improvement from structure 1 to 3. Similarly, ISC and CSC are significant for producing high-quality images. The performance increases with LDG, ISC, or CSC, and we can obtain improvements by using all of them. After adding HDMs, the performance increases to 31.93 dB compared to the structure 1 with 31.50 dB. This demonstrates the efficiency of our HDM in extracting informative features.

Multi-scale block. Finally, we show the effect of the multi-scale block based on the observations from Table 3. When MSBs are added, the PSNR value increases from 31.93 dB in structure 6 to 32.05 dB in structure 9. Comparisons between structure 1 and 2 or structure 4 and 7 demonstrate the effectiveness of the MSB as well. The performance improvements obtained by the MSB indicate that the multi-scale features have played an important part in generating SR images.

In Fig.7, we further visualize the training process of these nine structures on the PSNR of Set5 (×4). We can observe that the curves are consistent with our analyses, and the LDG, ISC, CSC, and MSB can further improve the performance. These quantitative and visual analyses show the superiority and effectiveness of our proposed LDG, ISC, CSC, and MSB elements.

Fig.8 shows comparisons of PSNR versus the number of parameters of our method and the compared methods. We can observe that our MS-RHDN and MS-RHDN+ have only half number of parameters of EDSR [7] and also fewer parameters than RDN [31]. They also achieve better performance. It demonstrates that our model has a more effective structure and a better trade-off between performance and model size.

VI. CONCLUSIONS
In this paper, we propose a novel multi-scale residual hierarchical dense network for high-quality image super-resolution. Our model aims at fully utilizing features in LR images. Specifically, the residual hierarchical dense structure is used for adaptively extracting multi-level features. Meanwhile, the multi-scale block serves to obtain multi-scale features. Furthermore, residual learning mechanism is used to stabilize the training of our model, and to pay attention to more informative features. Extensive experiments on benchmark datasets illustrate the effectiveness of our MS-RHDN in image super-resolution.

REFERENCES


***