Neurocomputing 398 (2020) 377–388

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Deep recursive up-down sampling networks for single image super-resolution

Zhen Li[a,c], Qilei Li[a], Wei Wu[a], Jinglei Yang[b], Zuoyong Li[c], Xiaomin Yang[a,c,*]

[a] College of Electronics and Information Engineering, Sichuan University, Chengdu, Sichuan 610064, China
[b] Department of Electrical and Computer Engineering, University of California, Santa Barbara, Santa Barbara, CA 93106, USA
[c] Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou 350121, China

ARTICLE INFO

Article history:
Received 30 August 2018
Revised 27 February 2019
Accepted 3 April 2019
Available online 25 April 2019

Keywords:
Single image super-resolution (SISR)
Deep learning
Recursive network
De-convolutional layer

ABSTRACT

Single image super-resolution (SISR) technology can reconstruct a high-resolution (HR) image from the corresponding low-resolution (LR) image. The emergence of deep learning pushes SISR to a new level. The successful application of the recursive network motivates us to explore a more efficient SISR method. In this paper, we propose the deep recursive up-down sampling networks (DRUDN) for SISR. In DRUDN, an original LR image is directly fed without extra interpolation. Then, we use the sophisticated recursive up-down sampling blocks (RUDB) to learn the complex mapping between the LR image and the HR image. At the reconstruction part, the feature map is up-scaled to the ideal size by a de-convolutional layer. Extensive experiments demonstrate that DRUDN outperforms the state-of-the-art methods in both subjective effects and objective evaluation.

1. Introduction

Limitations of optical sensors lead to the difficulty of obtaining an image with the ideal resolution. Single image super-resolution (SISR) technology can reconstruct a high-resolution (HR) image from the corresponding low-resolution (LR) image. SISR has been employed in various applications, such as surveillance [1], medicine [2] and object recognition [3]. However, since an LR image may be corrupted via various degradation operations, SISR is often regarded as an ill-posed problem. The key for SISR is to learn the mapping between LR and HR image pairs.

To reconstruct an HR image, numerous SISR methods have been developed. Among them, interpolation based methods are the most original and the simplest SISR methods. These methods require a little running time but also lead to blurring and jaggy artifacts near the edges of the recovered images. Moreover, to learn the mapping between LR and HR image pairs, some example-based methods are proposed. They can be roughly categorized into internal example-based methods [4–7] and external example-based methods [8–13]. Generally, internal example-based methods learn the mapping via the LR image itself, while external example-based methods learn the mapping via an external dataset. Some machine learning algorithms are widely employed for SISR, such as sparse representation [8,9], random forest [10], neighbor embedding [11], and neighborhood regression [12,13] These methods significantly improve the SISR performance. However, these are still some drawbacks remained to be settled, such as high computational costs, long running time and huge model size.

Recently, deep learning (DL), as an important branch of machine learning algorithms, has achieved great success in many computer vision tasks, such as object detection, image segmentation, face recognition and image retrieval [14–17]. DL also plays an important role in SISR as reviewed in [18].

Dong et al. (SRCNN) [19] first introduced the convolutional neural network (CNN) to SISR. In their method, a CNN with lightweight structure was employed to model the mapping so as to generate the recovered image. The effectiveness of their method demonstrates the feasibility of DL and provides a theoretical basis for the future works. Liu et. al [20] combined sparse prior with DL to generate the recovered image. This method also utilized the prior knowledge of the image, thus the recovered image can be obtained in a more robot way. It's a rule of thumb that the deeper network can obtain better performance. This was demonstrated by VGGnet [21] via winning the ImageNet image classification challenge [22]. Inspired by this, Kim et al. increased the network depth to 20 in VDSR [23]. By deepening the network, the performance is significantly improved compared with SRCNN. However, very deep neural network often faces gradient vanishing/exploding. To solve this problem, VDSR adopted gradient clipping. Besides, skip connections were employed, and the networks only needed to learn

* Corresponding author at: College of Electronics and Information Engineering, Sichuan University, Chengdu, Sichuan 610064, China.
E-mail address: arielyang@scu.edu.cn (X. Yang).

https://doi.org/10.1016/j.neucom.2019.04.004
0925-2312/

the residual component, rather than the whole image. Skip connections and gradient clipping not only reduce the possibility of gradient vanishing/exploding, but also accelerate the convergence of the network. ResNet [24] is an effective network architecture. It has been employed for SISR in many successful methods. Lim et al. proposed EDSR [25] by use ResNet with removing some unnecessary component to improve the performance and boost the SISR process. Even though, deep networks still require a lot of computational costs and running time. On this issue, Shi et al. [26] proposed a real-time SISR method by using a compact convolutional network model with the up-sample filter in the last layer to up-sample the output image into an ideal size. By doing so, the computational cost is efficiently reduced. Similarly, based on [19], Dong et al. [27] adopted deconvolution al layers with small convolutional kernel size to accelerate the SISR process. Another efficient way to reduce the computational cost is to use recursive blocks. Such as DRCN [28], DRRN [29], and DSRN [30]. These methods generally adopt a series of recursive blocks which are employed to learn the residual component of the recovered image. Due to the parameters are shared, these methods can reduce the parameter amount and avoid overfitting.

Generative adversary network (GAN) [31] provides another kind of solutions for SISR. Generally, the recovered image obtained via the generator is fed to discriminator, then discriminator will judge if the input image is an ground-truth image or an generated image. With such zero-sum game between generator and discriminator, the generated image can recover photo-realistic textures. Ledig et al. [32] first applied GAN to SISR. Wang et al. [33] used spatial feature transform layers following the GAN network to generate the recovered image with realistic and visually pleasing textures. Bulat and Tzimiropoulos proposed Super-FAN [34] which can accurately improve the resolution of LR face images. Recently, Wang et al. [35] won the PIRM2018-SR Challenge [36] using their ESRGAN, which well balances the trade-off between quantitative and perceptual qualities.

Loss function has a significant influence for DL based SISR methods. The L2 loss, also named mean square error loss, is widely deployed in some image restoration methods such as [19,27,37]. For SISR, L2 is to calculate the L2 distance (i.e. Euclidean distance) between generated image and HR image. However, using L2 often leads to some undesirable artifacts, which is not suitable for the human vision system (HVS). This is caused by that L2 assumes the noise is independent of the local characteristics. To reduce the artifacts introduced by L2 loss, the L1 loss can converge more fast, with an appropriate error penalization [38] which provides a significant improvement compared with the L2 loss. HVS is sensitive to local structural changes, thus structural similarity index (SSIM) can be used to measure the difference between the generated image and the HR image. Perceptual loss, as proposed by Johnson et al. [39], considers the correlations among high-level features. By using perception loss, the recovered image can be more realistic. Based on this, Ledig et al. [32] proposed the perceptual loss for GAN, in which the adversarial loss is also considered.

We propose a novel deep recursive up-down sampling networks (DRUDN) for SISR. Unlike DRRN, which accept the LR images obtained via bicubic interpolation as the input, we directly feed an original LR image to our network. Then, by passing a number of sophisticated recursive up-down sampling blocks (RUDB), a set of feature maps with same width and height as the input LR image are obtained. In the reconstruction part of our network, the LR image is up-scaled to the ideal size by a de-convolutional layer. By doing so, the reconstruction performance is significantly improved. To sum up, there are mainly two contributions:

1. We propose the deep recursive up-down sampling networks (DRUDN) for SISR. Unlike DRRN, which accept the LR images obtained via bicubic interpolations the input, we directly feed the original LR images to our network. At the reconstruction part, the feature maps are up-scaled to the ideal size by using a de-convolutional layer. Our network can obtain a highly accurate recovered image. It outperforms other state-of-the-art recursive SISR networks.
2. We propose recursive up-down sampling blocks (RUDB) to construct very deep trainable networks. Each RUDB consists of multiple groups of de-convolutional layers and convolutional layers. The de-convolutional layer can enlarge the size of features thus make the features more representative. Then, these features are further refined by the substantial convolutional layer. Such RUDB significantly improves the representational ability of the network.

The rest of the paper is organized as follow. Section 2 briefly reviews some representative recursive blocks structures and DRRN. Section 3 describes the proposed DRUDN in detail. Section 4 analyses the experimental results. Finally, Section 5 concludes the paper.

2. Related work

In this section, some classical recursive block structures are briefly reviewed at first. Besides, due to our method can be defined as an improvement of DRRN, thus we also briefly review the DRRN at second.

Recursive blocks are widely deployed in some state-of-the-art SISR methods, such as DRCN [28], DRRN [29], and DSRN [30]. The simplified recursive block structures of these methods are shown in Fig. 1. It's rule of thumb that deeper network can better use the characteristic of the features, thus a better performance can be achieved. These recursive networks are deepened by stacking a number of recursive blocks, thus it can better model the mapping between LR and HR image pairs. Besides, the weight sharing mechanism not only reduces the parameter amount but also avoid overfitting. As shown in Fig. 2, the architecture of these recursive networks can be generally divided into three parts, namely feature extraction part, recursive part, and reconstruction part.

2.1. Recursive block

Among the three parts, the recursive part plays the most important role since it enhances the representations of the input feature map by using the recursive blocks. Its working principle can be summarized as Eq. (1).

$$F_{out} = f^{B}(F_{in}), \tag{1}$$

where F_{in} and F_{out} are the input feature map fed to the recursive part and the corresponding output feature map from the recursive part, respectively. $f(\bullet)$ is the function of the recursive blocks. B is the number of the recursion (i.e. the number of the employed recursive blocks).

Next, some representative recursive block structures and the corresponding mathematical formulations will be simple reviewed. For all these mathematical formulations below, F_{in} denotes input feature map, and $f^{(n)}(F_{in})$ denotes the feature map after n recursions. For the sake of simplicity, we denote $f^{(n)}(F_{in})$ as f^{n}. Particularly, when $n = 0$, the f^{0} is equal to F_{in}, and all the biases are omitted.

DRCN: As shown in Fig. 1(a), the recursive block structure of DRCN consists of one convolutional layer with one ReLU [40]. Its mathematical formulation is shown in Eq. (2).

$$f^{n} = \varphi(f^{n-1}) = \sigma(Conv(W, f^{n-1})), \tag{2}$$

where $\varphi(\bullet)$ denotes the function of the recursive block of DRCN. Namely one convolution operation denoted as $Conv(\bullet)$ and one

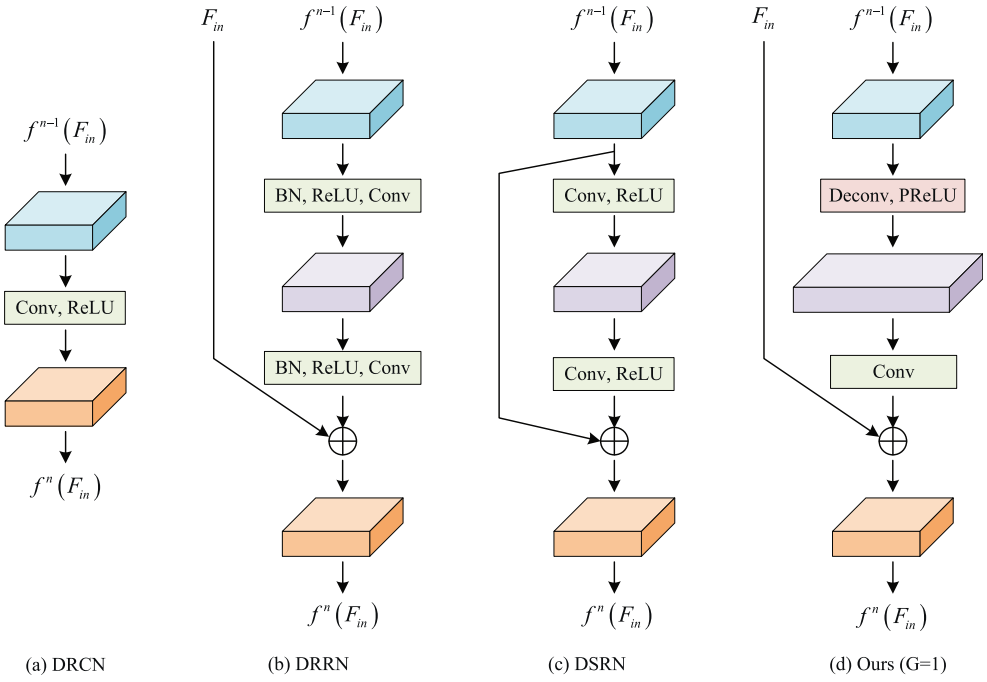


Fig. 1. Simplified recursive block structures of the DRCN, DRRN, DSRN, and our DRUDN.

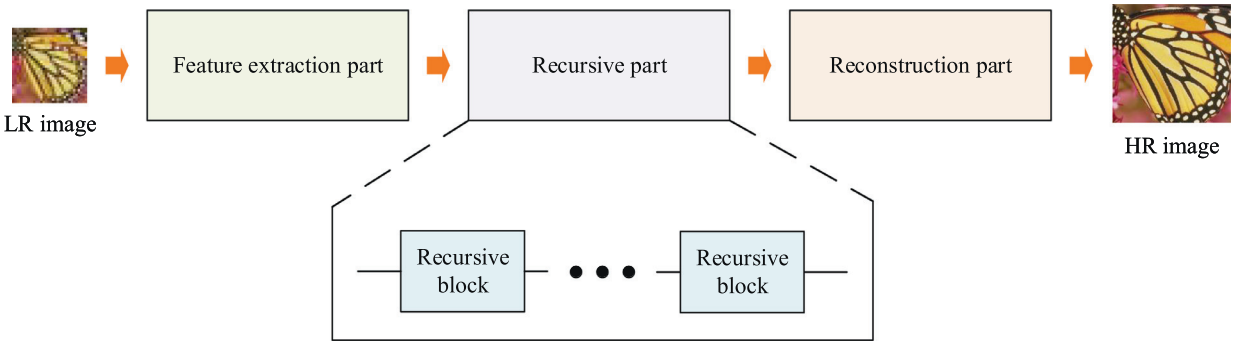


Fig. 2. Basic structure of recursive networks.

ReLU activation denoted as $\sigma(\bullet)$. W denotes the weight for the convolutional layer, and all the biases are omitted for simplicity. f^{n-1} denotes the output feature maps of $(n-1)$th recursion. f^n denotes the output of the next recursion, namely the output of nth recursion.

DRRN: As shown in Fig. 1(b), the recursive block structure of DRRN consists of two convolutional layers with two corresponding ReLUs. It should be noted that DRRN adopts pre-activation [41] structure, which means the activation operation is performed ahead of the convolutional layer. And batch normalization (BN) [42] is adopted before the activation units. Its mathematical formulation is shown in Eq. (3).

$$\begin{aligned} f^n &= f^0 + \varphi(f^{n-1}) \\ &= f^0 + Conv(W_2, \sigma(BN(Conv(W_1, \sigma(BN(f^{n-1})))))), \end{aligned} \tag{3}$$

where $\varphi(\bullet)$ denotes the function of recursive block of DRRN. Namely two convolution operation (with weight of W_1 and W_2, respectively) and two ReLU activation, $BN(\bullet)$ denotes the batch normalization operation.

DSRN: As shown in Fig. 1(c), same as DRRN, DSRN employs two convolutional layers with two corresponding ReLUs to form its recursive block. However, DSRN utilizes post-activation on the contrary with pre-activation. Whats more, another difference from DRRN is DSRN combines the output of the last iteration as the local residual, while DRRN combines the global input (i.e., f^0). Its mathematical formulation is shown in Eq. (4).

$$f^n = f^{n-1} + \varphi(f^{n-1}) = f^{n-1} + \sigma(Conv(W_2, \sigma(Conv(W_1, f^{n-1})))) \tag{4}$$

where $\varphi(\bullet)$ denotes the function of recursive block of DSRN.

2.2. DRRN

The main idea of DRRN is to deepen the network by stacking a series of elaborate recursive blocks. Compared to traditional chain mode, such design can prevent overfitting and reduce parameters. DRRN also aims to learn the residual component between LR image and HR image. As we described before, it can be roughly divided into three parts as shown in Fig. 2. Here, we will explain DRRN form these three aspects.

Feature extraction part: At the beginning of this part, the given LR image is first enlarged to ideal size by using bicubic interpretation. Then, a single convolutional layer is applied to extraction the shallow feature, which would be the input to the following part, namely recursive block parts. The feature extraction part can

be formulated as Eq. (5).

$$F_{in} = Conv(W_0, \sigma(x)), \tag{5}$$

where x is the interpolated input image, W_0 is the weight for the convolutional layer. F_{in} is the output feature map of the feature extraction part, also is the input of the recursive part.

Recursive part: The goal of this part is to learn the complex mapping between the LR image and the corresponding HR image by using these recursive blocks. This part is the core of the whole network. As shown in Fig. 1(b) the recursive block consists of two convolutional layers and two ReLUs. Besides, it employs F_{in} as the identity branch for all the unfolded recursive blocks. This is the difference between DRRN and other methods such as DSRN. This is a benefit to gradient back-propagation during training. Combining with the corresponding statement in Section 2.1, we can denote this part as Eq. (6).

$$F_{out} = f^B(F_{in}), \tag{6}$$

where f^B is the function of the recursive part with totally B recurrences. The detail function for the recurrence block can be seen in Eq. (3). F_{out} is the output feature map in this part.

Reconstruction part: Once F_{out} is obtained, the last part is to generate the recovered image. This part can be divided into two small steps. First is to generate the residual image. The F_{out} contains a lot of representative features. Similar to the feature extraction part, a single convolutional layer with a ReLU is employed to generate the learned residual. Finally, the recovered image can be obtained via combining the learned residual with the global residual, namely the interpolated input image. The part can be formulated as Eq. (7).

$$\begin{aligned} res &= Conv(W_{out}, \sigma(F_{out})), \\ SR &= x \oplus res, \end{aligned} \tag{7}$$

where W_{out} is the weight for the convolutional layer. res is the learned residual image. $\oplus$ denotes the element sum operation, and SR denotes the recovered image obtained by DRRN.

3. Deep recursive up-down sampling networks

In this section, we introduce our proposed deep recursive up-down sampling networks (DRUDN). Different from other recursive network, we design a sophisticated recursive block named RUDB. Such block employs a series of convolutional layer and de-convolutional layer pairs to enhance the representation ability of the feature. By doing so, the learned mapping between LR and HR is more robot and more reliable. Besides, rather than performing bicubic interpretation at the very beginning, we use a de-convolutional layer to upsample the residual image at the last step. This can efficiently improve the performance while maintaining the computational cost. As shown in Fig. 3, we also divide our DRUDN into three parts as other similar state-of-the-art methods do. The sophisticated recursive block is the core of our whole network. Logically, we first introduce RUDB. Then, the structure of DRUDN is presented in detail.

3.1. Recursive block

Here we introduce our recursive up-down sampling blocks named RUDB. It consists of a series of groups of de-convolutional layer and convolutional layer. Each group consists of one convolutional layer and one de-convolutional layer with the corresponding activation unit (except the last group which employs no activation function after the convolutional layer). For the sake of simplicity, we use a group as an example. The further study of the group(G) can be seen in Section 4.2. Besides, by stacking such recursive block for B times to form the recursive part, it is sufficient for extracting the representative features to learn complex mapping. Fig. 1 shows the structure of RUDB with G=1, and it is compared with the recursive blocks in some other recursive blocks. As Fig. 1 shows, given the input feature map F_{in}, which is also defined as the global residual, at the first recursive step, RUDB up-sample and down-sample the feature map for several times. With the changing the size, the feature map is more representative. When G=1, this step can be denoted as Eq. (8).

$$\begin{aligned} f^n &= F_{in} + \varphi(f^{n-1}) = f^0 + \varphi(f^{n-1}) \\ &= f^0 + Conv(W_2, \vartheta(Deconv(W_1, f^{n-1}))), \end{aligned} \tag{8}$$

where f^{n-1} denotes the inputs for the n_{th} recurrence. It should be noted that when $n = 1$, namely the first recurrence, the f^0 is equal to F_{in}. $\varphi(\bullet)$ is the function of RUDB. $Deconv(\bullet)$ and $Conv(\bullet)$ denotes the de-convolution operation and convolution operation, respectively. W_1 and W_2 are the weights for the de-convolutional layer and convolutional layer, respectively. $\vartheta(\bullet)$ refers to the PReLU [43] activation function.

From Fig. 1 and Eq. (8), it can be seen that our RUDB is similar to the recursive block of DRRN. However, the RUDB is an essential improvement. These are some great difference when compare RUDB with the recursive block of DRRN. First of all, their working mechanisms are different. The recursive block of DRRN learn the mapping via adopting a series of convolutional layers, while our RUDB employs de-convolutional layer and convolutional layer pairs to enhance the representation ability. Second, RUDB directly accepts the feature map with the same size as LR as its input, while DRRN accepts the feature map with the same size as HR as its input. In this matter, RUDB requires less computational cost. Besides, the learned feature is up-sampled by de-convolutional layer, this also can further improve the performance. Moreover, the recursive block of DRRN employs ReLU with 'pre-activation', while RUDB employs PReLU with normally 'post-activation'. In particular, for the last convolutional layer of RUDB, we use no activation function. This setting is based on the experimental results.

Moreover, our RUDB is inspired by DBPN [44]. However, in DBPN, the projection unit consists of an up-sample unit and an down-sample unit. The two sample units achieve the error feedback via computing the residual component. For every up-projection or down-projection unit, complicated calculation is required. By performing through numerous experiments, we argue that such over-complex design is completely unnecessary. (The detail performance comparison is can be seen in Section 4.2). Such design can only increase the complexity of the calculation, which has a side effect on the overall reconstruction effect. Thus, we totally redesign the up-sample unit and down-sample unit to form our RUDB. Compared with the projection unit of DBPN, our RUDB has two main benefits. First, we add residuals to the recursive unit, which facilitates the flow of gradients. Second, we can adopt multiple combinations to design a more flexible recursive block. Under the premise of ensuring the reconstruction performance, our block is more stable.

3.2. Network architecture

Analogous to other similar recursive SISR network, our DRUDN can also be divided into three parts, namely feature extraction part, recursive part, and reconstruction part. Each part will be described in detail in the rest of this section.

Feature extraction part: Given the input LR patch, to extract the shallow feature, a convolutional layer with size of $C_{in} \times 3 \times 3 \times C_f$ followed with PReLU is employed, where C_{in} is the number of channels for input images, C_f is a hyper-parameter denotes the number of channels of the output feature map. Here we use a filter with the size of 3×3 since it can take the adjacent

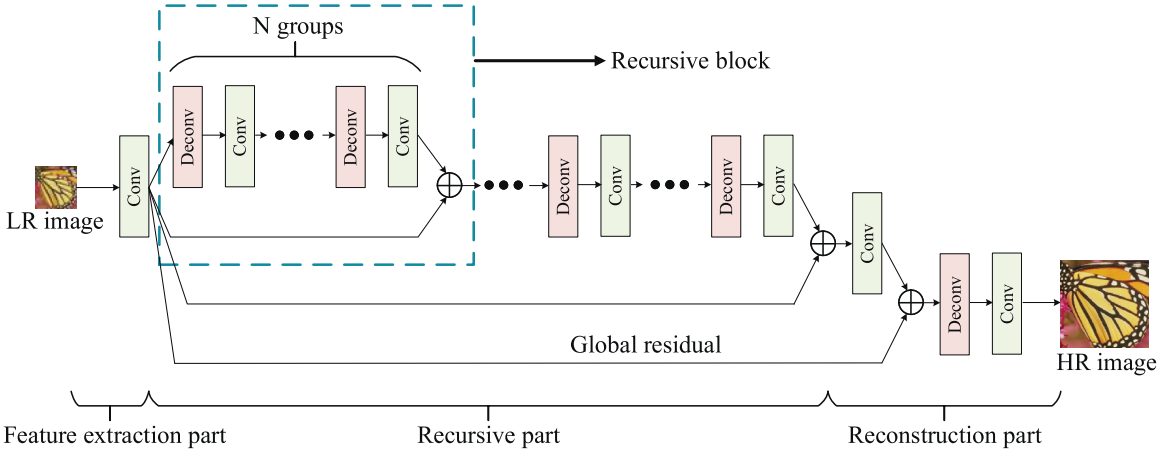


Fig. 3. Structure of deep recursive up-down sampling networks (DRUDN).

pixels into account. This part can be mathematically formulated as Eq. (9).

$$F_{in} = \vartheta(Conv(W_0, x)), \tag{9}$$

where x denotes the input LR image. $Conv(\bullet)$ and W_0 denotes the convolution operation and the corresponding weight, respectively. $\vartheta(\bullet)$ refers to the PReLU activation function. F_{in} is the output feature map, also is the input for the following recursive part.

Recursive part: This part plays the most important role in the whole network, since it accepts the output shallow feature map from the previous part, and recursively utilizes the representative characteristics in the feature map to mine the inner relationship between LR and HR pairs. To achieve this goal, we design a sophisticated recursive block named RUDB. Similar to DRRN, the identity branch for all the un-folded blocks is F_{in}. This is helpful for gradient flows during training. An entire recursive part consists B recursive blocks. All the B recursive blocks are connected in series. Each recursive block consists of G pairs of convolutional and deconvolutional layers. A network equipped with x recursive blocks, and each block has y up-down sampling groups is called 'GyBx'. All the unfolded blocks share the weight to reduce the parameter amount and to avoid overfitting. For the network with B recursive blocks, the entire process of this part can be summarized as Eq. (10).

$$F_{out} = f^B(F_{in}), \tag{10}$$

where f^B is the function of the recursive part with totally B blocks. The detail function for the recurrence block can be seen in Eq. 8. F_{out} is the output feature map in this part.

Reconstruction part: Assuming the output feature map from the previous part is size of $C_f \times H \times W$, where C_f is the channels number, and H and W are the height and width of the feature map, it also the height and width for the input LR image, since with appropriate padding strategy, H and W will not change during the recursive part. For this reconstruction part, the first is to refine the feature map convolutional layer with size of $C_f \times 3 \times 3 \times C_{in}$. Then is to combine the global residual with the F_{in} to obtain the intermediate feature (IF) via element-wise summation. The third is to upsample the intermediate feature to a bigger size, which equals the size of the HR image, via a de-convolutional layer. Finally, the upsampled intermediate feature is refined via a convolutional layer to be the recovered image. These four steps can be formulated as Eq. (11).

$$\begin{aligned} IF &= F_{in} + \vartheta(Conv(W_3, F_{out})), \\ SR &= Conv(W_5, \vartheta(Deconv(W_4, IF)))), \end{aligned} \tag{11}$$

where F_{out} is the output of the recursive part, MR is the intermediate image of the recovered image and the LR image. $Conv(\bullet)$, $Deconv(\bullet)$, $\vartheta(\bullet)$ and $\oplus$ are the convolution operation, deconvolution operation, PReLU, and element-wise summation operation, respectively. W_3, W_4 and W_5 are the weights for the corresponding convolutional layer or the de-convolutional layer, respectively.

Give the input LR image x, our goal of our network is to learn recovered image which is close to the ground truth image HR. We choose L1 loss to train our network. There are two main reasons for us to choose L1 loss rather than L2 loss. First, as described in [38,45], L2 loss is less suitable for human vision system of image quality. Human vision system is sensitive to texture and local structure in an image. However, L2 loss excessively penalties large errors, and tolerates small errors, destroying the underlying textures and local structures. On the contrary, L1 loss treats errors equally without discrimination, preserving abundant textures and local structures. Consequently, the model equipped with L1 loss can achieve a better reconstruction performance than L2 loss, which also has been proved in [25,38]. Second, as proved in [38], L1 loss can be easier to find a better minimum than L2 loss. Thus, L1 loss holds a better convergence process.

Therefore, for N training samples, the loss function can be formulated as

$$\Upsilon(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| HR^{(i)} - SR^{(i)} \right\|_1. \tag{12}$$

Moreover, we make a detailed comparison of the structure of DRUDN and DRRN as shown in Table 1. It can be seen that DRUDN removes unnecessary BN operations compared to DRRN and a more elaborate network is designed.

4. Experimental results and analysis

4.1. Implementation details

As introduced in Section 3.2, for feature extraction, we use a convolutional layer with the size of $C_{in} \times 3 \times 3 \times C_f$, where C_{in} and C_f are the channels number of input LR image and the output feature map. In this paper, since we use the image in RGB space to train and to test our network, thus C_{in} is set to 3. It's a rule of thumb that more feature channels can obtain better performance. However, it is proved by experiments that 64 feature channel is sufficient. Thus C_f is set to 64. For the recursive part, different settings are adopted for different scales as shown in Table 2. To a certain extent, the convolutional layer in the reconstruction part

Table 1
Detailed comparison of the structure of DRUDN and DRRN.

Method	Input image	Input image channel	Feature extraction part	Recursive part	Reconstruction part	Global residual location	Loss function
DRRN	LR+bicubic	Y	BN ReLU Conv	BN ReLU Conv	BN ReLU Conv Conv	HR space	L2
DRUDN	LR	RGB	Conv PReLU	Deconv PReLU Conv PReLU	PReLU Deconv PReLU Conv	LR space	L1

Table 2
The settings of RUDB in different scale.

Scale	Filter size	Stride	Padding
$\times 2$	6×6	2	2
$\times 3$	7×7	3	2
$\times 4$	8×8	4	2

can be regarded as a reverse of the one in the feature extraction part. The former turns the input LR image into C_{in} channels, and the latter turns the feature map into C_f channels. Thus the size of the convolutional layer in the reconstruction part is seen to $C_f \times 3 \times 3 \times C_{in}$. Besides, the size of the de-convolutional in the reconstruction part is also set in accordance with Table 2. The weights are initialized based on [43].

The proposed DRUDN is trained on DIV2K dataset with 800 images [46], which is commonly employed in recent literatures. We first crop each training image into image patches with the stride 240. Each image patch is of size 480×480. There are 32208 image patches totally. To form pairs of LR and HR image patches, we employ imresize function with the option bicubic in MATLAB to downsample each image patch. At training stage, we randomly crop a 32×32 RGB LR image sub-patch from an LR image patch at each iteration. At a corresponding position in an HR image patch, an HR image sub-patch can be obtained with a specified scale factor. Such a pair of LR and HR image sub-patches is randomly augmented (rotation or flipping), and is subtracted by RGB mean of DIV2K dataset. The batch size is set to 16. We use Adam [47] with the momentum of 0.9 as the optimizer. Our network is trained on 100 epochs. For G3B12 (the G and B will be explained in the following part), every epoch takes about 17 min. We use PyTorch [48] with an NVIDIA 1080 Ti GPU to implement our model. Bicubic down-sampling is employed to generate LR images for the test phase. For a fair comparison with other methods, we also perform shave operation according to the specified scale factor. For RGB images, the reconstruction results are tested on Y channel of YCbCr space. The evaluation metrics we used are PSNR and SSIM. The source code is available at https://github.com/liqilei/DRUDN.

4.2. Study of B and G

In this part, we first fix the recursive part to 48 layers, and combine the two key parameters, namely G and B in different ways. Then, we fix G to the 3 for investigating the influence of B in various combinations. We use the performance of VDSR [23] as a reference.

At first, we fix the unfolded recursive part with 48 layers (including convolutional layer and de-convolutional layer). Due to each G consists two layer (a convolutional layer and a de-convolutional layer), thus the combinations can be G1B24, G2B12, G3B8, G4B6. We test their performances in Set5 and Set14 with $\times 4$ scale factor and the result is shown in Table 3. Since the running time only has a minor change when the number of parameter increases, we only take the parameters comparison as a consideration. From Table 3, we can see that PSNR value first increases and then decays a little as the network parameters increasing. This reflects the over-fitting problem caused by redundant parameters. Besides, we also fix the unfolded recursive part of DBPN with 48 layers to compare the performance. It can be seen that our network outperforms DBPN unit[44] while maintains the time and space complexity. The convergence process of these 5 combinations is shown in Fig. 4.

Secondly, to study the influence of B, we fix G = 3. The number of network parameters thus retains 1445 K. Experiments are performed on different combinations, namely G3B6, G3B8, G3B10, G3B12. The average PSNR of these combinations is shown in Table 4, and the visualization of the convergence is shown in Fig. 5. From Table 4, it can be seen that with the increase of B, the overall performance get a slight improvement and the execution time

Table 3
Running time, model parameters and PSNR analysis for $\times 4$ on datasets Set5 for various combinations and one projection unit in DBPN [44] of 48 layers.

Dataset	G1B24	G2B12	G3B8	G4B6	DBPN unit
Parameters (K)	827	1352	1445	1876	1445
Running time (s)	0.011	0.013	0.013	0.015	0.014
PSNR (dB)	32.07	32.14	**32.15**	32.13	32.13

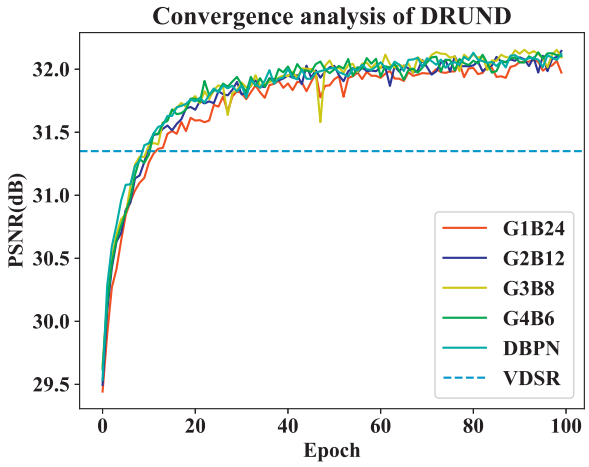


Fig. 4. Convergence visualization of 48 layers with scale factor $\times 4$ on Set5.

Table 4
Average PSNR for $\times 4$ on datasets Set5 for various combinations of G=3 (1,445K parameters).

Dataset	G3B6	G3B8	G3B10	G3B12
Running time (s)	0.013	0.013	0.015	0.016
PSNR (dB)	32.15	32.15	32.12	**32.17**

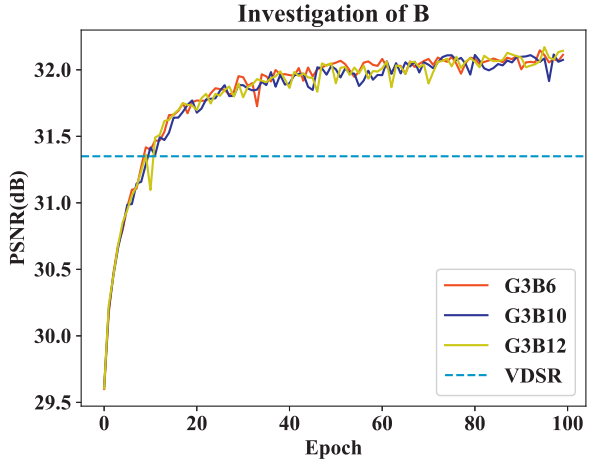


Fig. 5. Study the influence of B in the case of G=3 with scale factor $\times 4$ on Set5.

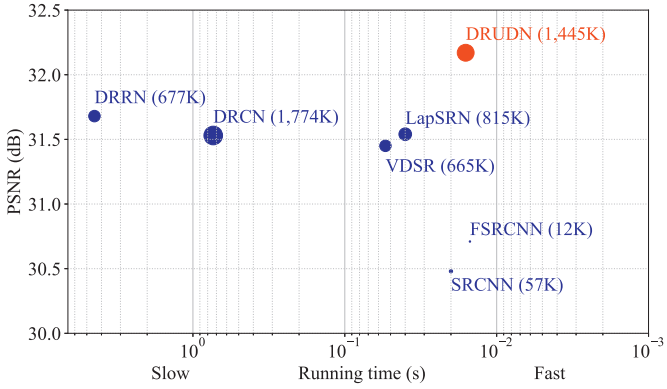


Fig. 6. Running time, the number of parameters and accuracy trade-off. The results are evaluated on Set5 with scale factor $\times 4$. The bigger marker represents the larger number of parameters.

remains roughly the same. This is because our DRUDN has strong representation capabilities and does not require complex structures to meet the requirements of SISR task.

4.3. Model analysis

We compare the running time and model parameters on the computer with 3.5 GHz Intel I7 (32G RAM) and an NVIDIA 1080 Ti GPU. These comparisons only consider deep learning based methods. To make a fair running time comparison, we re-implement SRCNN and FSRCNN on GPU which are only provided CPU version officially. Other contrasting methods are evaluated using their official codes. The trade-off among running time, the number of parameters and accuracy is shown in Fig. 6. Although our proposed method holds a large number of parameters, the running time of our proposed method is faster than other deep learning based methods besides FSRCNN. The reason is that the computational cost will dramatically decrease by taking the original LR image as input instead of interpolating the LR image firstly [27]. More importantly, the reconstruction performance of our proposed method surpasses other compared methods by a large margin.

4.4. Comparison with the-state-of-the-arts on gray image datasets

We first evaluate our proposed method on three gray image datasets: (a) natural image dataset, (b) medical image dataset, and (c) infrared image dataset. The natural image dataset (containing 19 images) is obtained by transforming Set5 and Set14 from RGB space to gray space. The medical image dataset (containing 13 images) is collected from SIEMENS official website[1]. The infrared image dataset (containing 49 images) is collected from TNO image fusion dataset [49]. The compared state-of-the-art methods including A+[12], SRCNN [19] and DRRN [29]. We use G3B12 model for this comparison. The quantitative and qualitative results are shown in Table 5 and Fig. 7, respectively.

The reconstruction results on these gray images exhibit the vigorous power of DRUDN in recovering the texture. The proposed DRUDN can restores the detail accurately, while other methods often lead to unsatisfying results with blurred detail or incorrect text direction. For instance, A+ and SRCNN fail to generate faithful reconstruction images. Even worse, the direction of the enlarged texture in their 'barbara' image and the 'Reek' image are totally wrong. The proposed DRUDN can learn the mapping between the LR image and the HR image to precisely construct the SR image, which can be very close to the ground truth image. In other word, DRUDN can accurately recover the texture detail in nature image and the thermal radiation information in infrared image. These evidences indicate that our DRUDN can obtain more representative information than other methods. The quantitative results shown in Table 5 reveals that out DRUDN achieve the top performance, which demonstrate the effectiveness of our DRUDN on the gray image SISR.

4.5. Comparison with the-state-of-the-arts on RGB image datasets

The proposed method is compared with some state-of-the-art methods, including A+ [12], SRCNN [19], FSRCNN [27], SelfExSR [4], RFL [10], SCN [50] VDSR [23], DRCN [28], LapSRN [51], DRRN [29], and DSRN [30]. The proposed method is tested on four standard RGB benchmark datasets, namely Set5 [11] with 5 images, Set14 [52] with 14 images, B100 [53] with 100 images, and Urban100 [4] with 100 images. The result of these contrast experiments are obtained from [30]. We use G3B12 model for this comparison. **Quantitative evaluation**: The average PSNR/SSIM for scale factor $\times 2$, $\times 3$, $\times 4$ on Set5, Set14, B100 dataset of DRUDN with other contrast methods is shown in Table 6. The best results are shown in **bold** and the second best results are underlined. It can be seen that our method has an obvious superiority in objective evaluation indicators. Our DRUDN can generate satisfying results for all the different scales. Moreover, one may notice that the advantages at small scales (i.e., 2) are not obvious when compared with the results in larger scales (i.e., 3, 4). This is because the DRUDN upsamples the recovered image to the ideal size by using the deconvolutional layer at the end, rather than performing bicubic interpolation at the very beginning as other methods do. In detail, on a small scale, bicubic interpolation can achieve better results than deconvolution. But on a large scale, interpolation will lose a lot of detail, causing the image to be blurred. Which de-convolution layer generate a better recovered image through the complex mapping. By comparing the data in Table 6, the superiority of our method is demonstrated.

Subjective effect: Part of the recovered image as shown in Fig. 8. First, we show the recovered results for 'img_005' in Urban100 dataset, we can see that our method can generate a more faithful recovered image which can recover most sharp lines. The outline of the windows is the closest to the HR image, and no any other extra structure is introduced. However, other contrast methods fail to recover the detail information. What worse, DRRN produces obvious artifacts around the boundary of the windows. The second group image named 'comic' comes from Set14. The

[1] https://www.healthcare.siemens.com/.

Table 5
Average PSNR/SSIM for scale factor $\times 4$ on natural, medical and infrared image datasets of DRUDN with other contrast methods. The best results are shown in **bold**.

Method	Natural image dataset PSNR/SSIM	Medical image dataset PSNR/SSIM	Infrared image dataset PSNR/SSIM
Bicubic	25.85/0.698	28.58/0.831	34.94/0.884
A+	28.18/0.780	32.07/0.895	37.37/0.912
SRCNN	27.09/0.739	30.27/0.861	36.06/0.896
DRRN	29.19/0.804	32.65/0.906	37.83/0.916
DRUDN	**29.34**/**0.806**	**33.05**/**0.908**	**37.99**/**0.917**

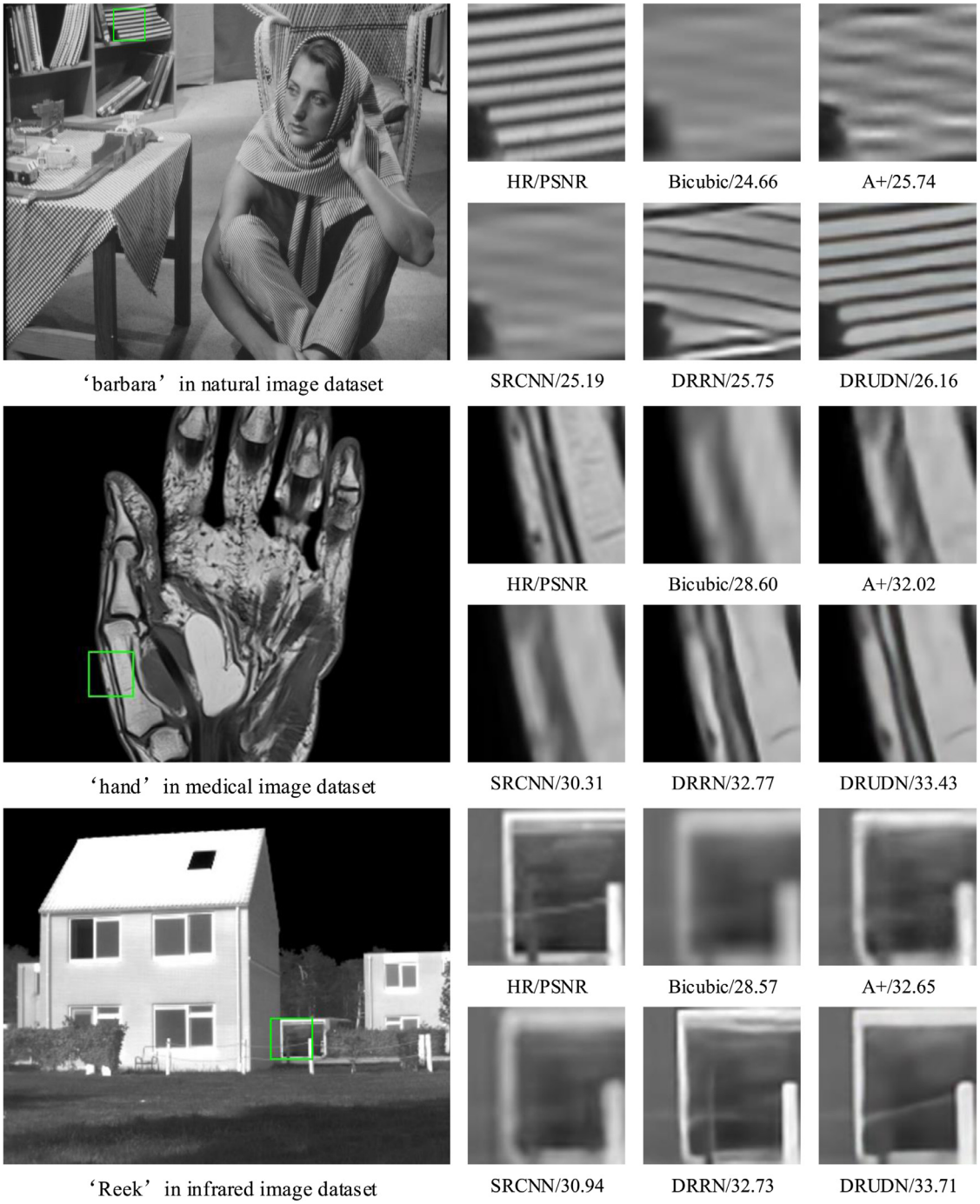


Fig. 7. Qualitative comparison of DRUDN with other state-of-the-art methods on gray image datasets.

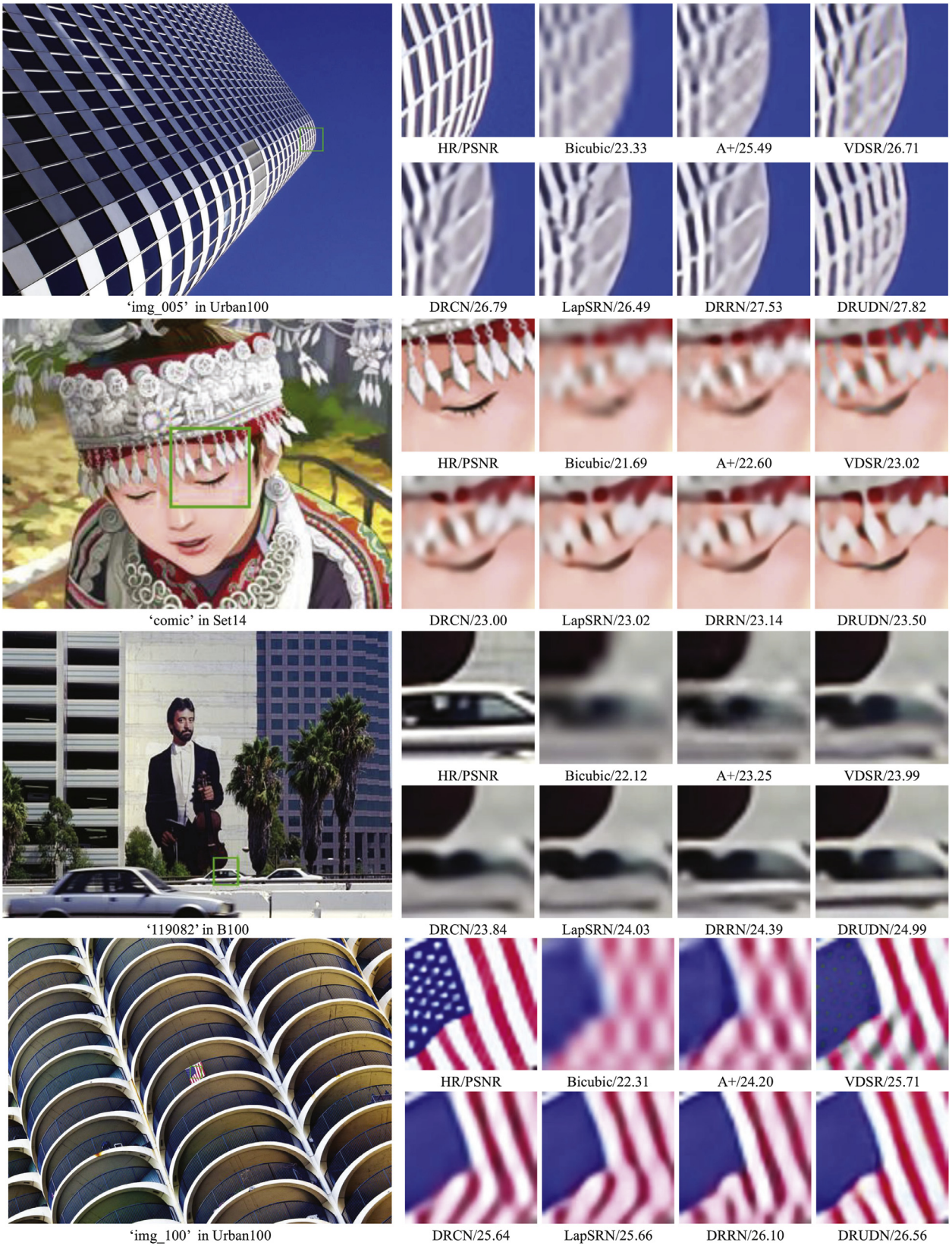


Fig. 8. Qualitative comparison of DRUDN with other state-of-the-art methods on $\times 4$ SISR.

Table 6
Benchmark results. Average PSNR/SSIM for scale factor $\times 2$, $\times 3$, $\times 4$ on Set5, Set14, B100, and Urban100 datasets of DRUDN with other contrast methods. The best results are shown in **bold** and the second best results are underlined.

Method	Scale	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM
Bicubic	2	33.65 / 0.930	30.34 / 0.870	29.56 / 0.844	26.88 / 0.841
A+	2	36.54 / 0.954	32.40 / 0.906	31.22 / 0.887	29.23 / 0.894
SRCNN	2	36.65 / 0.954	32.29 / 0.903	31.36 / 0.888	29.52 / 0.895
FSRCNN	2	36.99 / 0.955	32.73 / 0.909	31.51 / 0.891	29.87 / 0.901
SelfExSR	2	36.49 / 0.954	32.44 / 0.906	31.18 / 0.886	29.54 / 0.897
RFL	2	36.55 / 0.954	32.36 / 0.905	31.16 / 0.885	29.13 / 0.891
SCN	2	36.52 / 0.953	32.42 / 0.904	31.24 / 0.884	29.50 / 0.896
VDSR	2	37.53 / 0.958	32.97 / 0.913	31.90 / 0.896	30.77 / 0.914
DRCN	2	37.63 / 0.959	32.98 / 0.913	31.85 / 0.894	30.76 / 0.913
LapSRN	2	37.52 / 0.959	33.08 / 0.913	31.80 / 0.895	30.41 / 0.910
DRRN	2	**37.74** / 0.959	33.23 / 0.914	32.05 / 0.897	31.23 / 0.919
DSRN	2	37.66 / 0.959	33.15 / 0.913	**32.10** / 0.897	30.97 / 0.916
DRUDN	2	37.68 / **0.959**	**33.31** / **0.915**	32.02 / **0.897**	**31.53** / **0.922**
Bicubic	3	30.39 / 0.868	27.64 / 0.776	27.21 / 0.740	24.46 / 0.736
A+	3	32.60 / 0.908	29.24 / 0.821	28.30 / 0.784	26.05 / 0.798
SRCNN	3	32.76 / 0.908	29.41 / 0.823	28.41 / 0.787	26.24 / 0.800
FSRCNN	3	33.15 / 0.913	29.53 / 0.826	28.52 / 0.790	26.42 / 0.807
SelfExSR	3	32.63 / 0.908	29.33 / 0.823	28.29 / 0.785	26.45 / 0.809
RFL	3	32.45 / 0.905	29.15 / 0.819	28.22 / 0.782	25.87 / 0.791
SCN	3	32.60 / 0.907	29.24 / 0.819	28.32 / 0.782	26.21 / 0.801
VDSR	3	33.66 / 0.921	29.77 / 0.834	28.83 / 0.798	27.14 / 0.829
DRCN	3	33.82 / 0.922	29.76 / 0.833	28.80 / 0.797	27.15 / 0.828
LapSRN	3	33.78 / 0.921	29.87 / 0.833	28.81 / 0.797	27.06 / 0.827
DRRN	3	34.03 / 0.924	29.96 / 0.835	28.95 / 0.800	27.53 / 0.838
DSRN	3	33.88 / 0.922	**30.26** / 0.837	28.81 / 0.797	27.16 / 0.828
DRUDN	3	**34.25** / **0.925**	30.20 / **0.838**	**29.01** / **0.802**	**27.89** / **0.846**
Bicubic	4	28.42 / 0.810	26.10 / 0.704	25.96 / 0.669	23.15 / 0.659
A+	4	30.30 / 0.859	27.43 / 0.752	26.82 / 0.710	24.34 / 0.720
SRCNN	4	30.49 / 0.862	27.61 / 0.754	26.91 / 0.712	24.53 / 0.724
FSRCNN	4	30.71 / 0.865	27.70 / 0.756	26.97 / 0.714	24.61 / 0.727
SelfExSR	4	30.33 / 0.861	27.54 / 0.756	26.84 / 0.712	24.82 / 0.740
RFL	4	30.15 / 0.853	27.33 / 0.748	26.75 / 0.707	24.20 / 0.711
SCN	4	30.39 / 0.862	27.48 / 0.751	26.87 / 0.710	24.52 / 0.725
VDSR	4	31.35 / 0.882	28.03 / 0.770	27.29 / 0.726	25.18 / 0.753
DRCN	4	31.53 / 0.884	28.04 / 0.770	27.24 / 0.724	25.14 / 0.752
LapSRN	4	31.54 / 0.885	28.19 / 0.772	27.32 / 0.728	25.21 / 0.756
DRRN	4	31.68 / 0.889	28.21 / 0.772	27.38 / 0.728	25.44 / 0.764
DSRN	4	31.40 / 0.883	28.07 / 0.770	27.25 / 0.724	25.08 / 0.747
DRUDN	4	**32.17** / **0.894**	**28.56** / **0.780**	**27.54** / **0.734**	**25.99** / **0.783**

shape of the headgear is well restored by our DRUDN. Although the recovered results of the VDSR also restores the general outline, the color has changed. For '119082' in B100, our method can generate the clearest recovered image which is the closest to the HR image. From the enlarged images of 'img_100', the advantages of our DRUDN are clearly demonstrated. In the recovered images obtained from A+, DRCN, LapSRN, DRRN, the direction of the flag is wrong. What's more, although the VDSR has the correct direction, it has serious discoloration and artificial effects, which is unacceptable. However, the recovered image obtained by DRUDN better restores the original details and does not introduce other structures. This demonstrates that DRUDN can generate subjectively more satisfactory recovered images.

5. Conclusion

In this paper, we propose a novel deep recursive up-down sampling networks (DRUDN) for SISR. We design a sophisticated recursive up-down sampling block (RUDB) to improves the representational ability of the network. In the reconstruction part of our network, the LR image is up-scaled to the ideal size by a de-convolutional layer. By doing so, the features are more representative. Then, these features are further refined by the substantial convolutional layer. Extensive experiments demonstrate that DRUDN outperforms the state-of-the-art methods in both subjective effects and objective evaluation.

Conflict of interest

None declared.

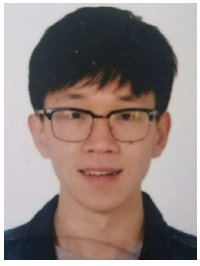
Acknowledgment

The research in our paper is sponsored by National Natural Science Foundation of China (No.61701327), Science Foundation of Sichuan Science and Technology Department (No. 2018GZ0178), Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) (No. MJUKF-IPIC201805).

References

[1] W.W. Zou, P.C. Yuen, Very low resolution face recognition problem, IEEE Trans. Image Process. 21 (1) (2012) 327–340.
[2] W. Shi, J. Caballero, C. Ledig, X. Zhuang, W. Bai, K. Bhatia, A.M.S.M. de Marvao, T. Dawes, D. ORegan, D. Rueckert, Cardiac image super-resolution with global correspondence using multi-atlas patchmatch, in: Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, 2013.
[3] M.S. Sajjadi, B. Schölkopf, M. Hirsch, Enhancenet: single image super-resolution through automated texture synthesis, in: Proceedings of the ICCV, 2017.
[4] J.-B. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, in: Proceedings of the CVPR, 2015.

[5] C.-Y. Yang, J.-B. Huang, M.-H. Yang, Exploiting self-similarities for single frame super-resolution, in: Proceedings of the ACCV, 2010.
[6] D. Glasner, S. Bagon, M. Irani, Super-resolution from a single image, in: Proceedings of the ICCV, 2009.
[7] W. Wu, C. Zheng, Single image super-resolution using self-similarity and generalized nonlocal mean, in: Proceedings of the TENCON, 2013.
[8] J. Yang, Z. Wang, Z. Lin, S. Cohen, T. Huang, Coupled dictionary training for image super-resolution, IEEE Trans. Image Process. 21 (8) (2012) 3467–3478.
[9] J. Yang, J. Wright, T.S. Huang, Y. Ma, Image super-resolution via sparse representation, volume 19, 2010, pp. 2861–2873.
[10] S. Schulter, C. Leistner, H. Bischof, Fast and accurate image upscaling with super-resolution forests, in: Proceedings of the CVPR, 2015.
[11] M. Bevilacqua, A. Roumy, C. Guillemot, M.L. Alberi-Morel, Low-complexity single-image super-resolution based on nonnegative neighbor embedding, in: Proceedings of the BMVC, 2012.
[12] R. Timofte, V. De Smet, L. Van Gool, A+: Adjusted anchored neighborhood regression for fast super-resolution, in: Proceedings of the ACCV, 2014.
[13] R. Timofte, V. De Smet, L. Van Gool, Anchored neighborhood regression for fast example-based super-resolution, in: Proceedings of the ICCV, 2013.
[14] J. Dai, Y. Li, K. He, J. Sun, R-FCN: object detection via region-based fully convolutional networks, in: Proceedings of the NeurIPS, 2016.
[15] E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (4) (2017) 640–651.
[16] O.M. Parkhi, A. Vedaldi, A. Zisserman, Deep face recognition, in: Proceedings of the BMVC, 2015.
[17] R. Tavoli, E. Kozegar, M. Shojafar, H. Soleimani, Z. Pooranian, Weighted PCA for improving document image retrieval system based on keyword spotting accuracy, in: Proceedings of the International Conference on Telecommunications and Signal Processing, 2013.
[18] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, Deep learning for single image super-resolution: a brief review, arXiv:1808.03344 (2018).
[19] C. Dong, C.C. Loy, K. He, X. Tang, Learning a deep convolutional network for image super-resolution, in: Proceedings of the ECCV, 2014.
[20] D. Liu, Z. Wang, B. Wen, J. Yang, W. Han, T.S. Huang, Robust single image super-resolution via deep networks with sparse prior, IEEE Trans. Image Process. 25 (7) (2016) 3194–3207.
[21] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 (2014).
[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. (IJCV) 115 (3) (2015) 211–252.
[23] J. Kim, J. Kwon Lee, K. Mu Lee, Accurate image super-resolution using very deep convolutional networks, in: Proceedings of the CVPR, 2016.
[24] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the CVPR, 2016.
[25] B. Lim, S. Son, H. Kim, S. Nah, K.M. Lee, Enhanced deep residual networks for single image super-resolution, in: Proceedings of the CVPRW, 2017.
[26] W. Shi, J. Caballero, F. Huszár, J. Totz, A.P. Aitken, R. Bishop, D. Rueckert, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: Proceedings of the CVPR, 2016.
[27] C. Dong, C.C. Loy, X. Tang, Accelerating the super-resolution convolutional neural network, in: Proceedings of the ECCV, 2016.
[28] J. Kim, J. Kwon Lee, K. Mu Lee, Deeply-recursive convolutional network for image super-resolution, in: Proceedings of the CVPR, 2016.
[29] Y. Tai, J. Yang, X. Liu, Image super-resolution via deep recursive residual network, in: Proceedings of the CVPR, 2017.
[30] W. Han, S. Chang, D. Liu, M. Yu, M. Witbrock, T.S. Huang, Image super-resolution via dual-state recurrent networks, in: Proceedings of the CVPR, 2018.
[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the NeurIPS, 2014.
[32] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: Proceedings of the CVPR, 2017.
[33] X. Wang, K. Yu, C. Dong, C.C. Loy, Recovering realistic texture in image super-resolution by deep spatial feature transform, 2018.
[34] A. Bulat, G. Tzimiropoulos, Super-fan: integrated facial landmark localization and super-resolution of real-world low resolution faces in arbitrary poses with gans, 2018.
[35] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, C.C. Loy, Esrgan: enhanced super-resolution generative adversarial networks, in: Proceedings of the ECCVW, 2018.
[36] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, L. Zelnik-Manor, The 2018 PIRM challenge on perceptual image super-resolution, in: Proceedings of the ECCVW, 2018.
[37] Y.-Q. Wang, A multilayer neural network for image demosaicking, in: Proceedings of the ICIP, 2014.
[38] H. Zhao, O. Gallo, I. Frosio, J. Kautz, Loss functions for image restoration with neural networks, IEEE Trans. Comput. Imaging 3 (1) (2017) 47–57.
[39] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: Proceedings of the ECCV, 2016.
[40] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the ICML, 2010.
[41] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: Proceedings of the ECCV, 2016.
[42] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv:1502.03167 (2015).
[43] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the ICCV, 2015.
[44] M. Haris, G. Shakhnarovich, N. Ukita, Deep back-projection networks for super-resolution, in: Proceedings of the CVPR, 2018.
[45] Z. Wang, A.C. Bovik, Mean squared error: love it or leave it? A new look at signal fidelity measures, IEEE Signal Process. Mag. 26 (1) (2009) 98–117.
[46] E. Agustsson, R. Timofte, NTIRE 2017 challenge on single image super-resolution: dataset and study, in: Proceedings of the CVPRW, 2017.
[47] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).
[48] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).
[49] A. Toet, TNO image fusion dataset (2014). doi:10.6084/m9.figshare.1008029.v1.
[50] Z. Wang, D. Liu, J. Yang, W. Han, T. Huang, Deep networks for image super-resolution with sparse prior, in: Proceedings of the ICCV, 2015.
[51] W.-S. Lai, J.-B. Huang, N. Ahuja, M.-H. Yang, Deep Laplacian pyramid networks for fast and accurate superresolution, in: Proceedings of the CVPR, 2017.
[52] R. Zeyde, M. Elad, M. Protter, On single image scale-up using sparse-representations, in: Proceedings of the International Conference on Curves and Surfaces, 2010.
[53] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proceedings of the ICCV, 2001.



Zhen Li is currently pursuing the M.S. degree in college of electronics and information engineering from Sichuan University, Chengdu, China. His research interests are image restoration and deep learning.

Qilei Li is currently pursuing the M.S. degree in college of electronics and information engineering from Sichuan University, Chengdu, China. He received his BS degrees in electronic information engineering from Shandong University of Technology. His research interests are image processing and deep learning.

Wei Wu is currently a professor in College of Electronics and Information Engineering, Sichuan University. He received his BS degree from Tianjin University, and received his MS and PhD degrees in communication and information system from Sichuan University. He worked in a National Research Council Canada as a post doctorate for one year. His research interests are image processing and pattern recognition.

Jinglei Yang is currently a master's student in electrical and computer engineering at University of California, Santa Barbara. She received her bachelor's degree in electronics and information engineering from Sichuan University in China in 2018. Her research interests are image processing and deep learning.

Zuoyong Li received the B.S. and M.S. degrees in Computer Science and Technology from Fuzhou University, Fuzhou, China, in 2002 and 2006, respectively. He received the Ph.D. degree from the School of Computer Science and Technology at Nanjing University of Science and Technology, Nanjing (NUST), China, in 2010. He is currently a professor in Department of Computer Science of Minjiang University, Fuzhou, China. He has published over 60 papers in international/national journals. His current research interest is image processing, pattern recognition and machine learning.

Xiaomin Yang is currently an associate professor in College of Electronics and Information Engineering, Sichuan University. She received her BS degree from Sichuan University, and received her PhD degree in communication and information system from Sichuan University. She worked in University of Adelaide as a post doctorate for one year. Her research interests are image processing and pattern recognition.