Lightweight and Efficient
Image Super-Resolution
with Block State-based Recursive Network

Jun-Ho Choi, Jun-Hyuk Kim, Manri Cheon, and Jong-Seok Lee

School of Integrated Technology, Yonsei University, Korea
{idearibosome, junhyuk.kim, manri.cheon, jong-seok.lee}@yonsei.ac.kr

Abstract. Recently, several deep learning-based image super-resolution methods have been developed by stacking massive numbers of layers. However, this leads too large model sizes and high computational complexities, thus some recursive parameter-sharing methods have been also proposed. Nevertheless, their designs do not properly utilize the potential of the recursive operation. In this paper, we propose a novel, lightweight, and efficient super-resolution method to maximize the usefulness of the recursive architecture, by introducing block state-based recursive network. By taking advantage of utilizing the block state, the recursive part of our model can easily track the status of the current image features. We show the benefits of the proposed method in terms of model size, speed, and efficiency. In addition, we show that our method outperforms the other state-of-the-art methods.

Keywords: Super-resolution, deep learning, recursive neural network

1 Introduction

Single-image super-resolution is a task to obtain a high-resolution image from a given low-resolution image. It is a kind of ill-posed problems since it has to estimate image details under the lack of spatial information. Many researchers have proposed various approaches that can generate upscaled images having better quality than the simple interpolation methods such as nearest-neighbor, bilinear, and bicubic upscaling.

Recently, the emergence of deep learning techniques has flowed into the super-resolution field. For example, Dong *et al.* [8] proposed the super-resolution convolutional neural network (SRCNN) model, which showed much improved performance in comparison to the previous approaches. Lim *et al.* [19] suggested the enhanced deep super-resolution (EDSR) model, which employs residual connections and various optimization techniques.

Many recent deep learning-based super-resolution methods tend to stack much more numbers of layers to obtain better upscaled images, but this dramatically increases the number of involved model parameters. For instance, the EDSR model requires about 43M parameters, which are at least 400 times more
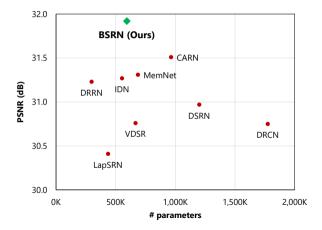
Fig. 1. Number of parameters and peak signal-to-noise ratio (PSNR) values of the state-of-the-art and the proposed methods for an upscaling factor of 2 on the Urban100 dataset [11].

than those of the SRCNN model. To deal with this, recursive approaches that use some parameters repeatedly have been proposed, including deeply-recursive convolutional network (DRCN) [14], deep recursive residual network (DRRN) [24], and dual-state recurrent network (DSRN) [9].

The recursive super-resolution methods can be regarded as kinds of recurrent neural networks (RNNs) [9]. RNNs have been usually employed when sequential relation of the data is significant, such as language modeling [26] and human activity recognition [7]. The beauty of RNNs comes from their two-fold structure: the recurrent unit handles not only the current input data but also the previously processed features. Since the previously processed features contain historical information, RNNs can deal with sequential dependency of the inputs properly.

However, two characteristics of the existing recursive super-resolution methods hinder them from fully exploiting the usefulness of the RNNs. First, there are no intermediate inputs and only the previously processed features are provided to the recurrent unit. Second, the final output of the recurrent unit is directly used to obtain the final upscaled image. In this situation, the output of the recurrent unit has to contain not only the super-resolved features, but also the historical information that is not useful in the non-recursive post-processing part.

To alleviate this problem, we propose a novel super-resolution method using block state-based recursive network (BSRN). Our method employs so-called "block state" along with the input features in the recursive part, which is a separate information storage to keep historical features. Thanks to the elaborate design, our method achieves various benefits on top of the previous recursive super-resolution methods, in terms of image quality, lightness, speed, and effi-

ciency. As shown in Fig. 1, our method achieves the best performance in terms of image quality, while the model complexity is significantly reduced. In addition, the BSRN model can generate the super-resolved images in a progressive manner, which is useful for real-world applications such as progressive image loading.

The rest of the paper is organized as follows. First, we discuss the related work in Section 2. Then, the overall structure of the proposed method is explained in Section 3. We present several experiments for in-depth analysis of our method in Section 4, including examining effectiveness of the newly introduced recursive structure and comparison with the other state-of-the-art methods. Finally, we conclude our work in Section 5.

2   Related Work

Before deep learning has emerged, feature extraction-based methods have been widely used for super-resolution, such as sparse representation-based [28] and Bayes forest-based [22] approaches. This trend has changed since deep learning showed significantly better performance in image classification tasks [17]. Dong *et al.* [8] pioneered the deep learning-based super-resolution by introducing SR-CNN, which enhances the interpolated image via three convolutional layers. Kim *et al.* [13] proposed very deep super-resolution (VDSR), which stacks 20 convolutional layers to improve the performance. Lim *et al.* [19] suggested the EDSR model, which employs more than 64 convolutional layers. These methods share the basic empirical rule of deep learning: deeper and larger models can achieve better performance [21].

As we addressed in the introduction, super-resolution methods sharing model parameters have been proposed. DRCN introduced by Kim *et al.* [14] proves the effectiveness of parameter sharing, which recursively applies the feature extraction layer for 16 times. Tai *et al.* [24] proposed DRRN that employs residual network (ResNet) [10] with sharing the model parameters. They also proposed the memory network (MemNet) model [25], which contains groups of recursive parts called "memory blocks" with skip connections across them. Han *et al.* [9] considered DRCN and DRRN as the RNNs employing recurrent states, and proposed DSRN, which uses dual recurrent states. Ahn *et al.* [3] developed the cascading residual network (CARN) model, which employs cascading residual blocks with sharing their model parameters. Although these methods can be regarded as RNNs as Han *et al.* mentioned [9], none of them uses a separate state, which is used in only the recursive part and not in the non-recursive post-processing part.

Some researchers proposed super-resolution methods that do not rely on shared parameters but have small numbers of model parameters. For example, Lai *et al.* [18] introduced the Laplacian pyramid super-resolution network (LapSRN) method, which progressively upscales the input image by a factor of 2. Hui *et al.* [12] proposed the information distillation network (IDN) method, which employs long and short feature extraction paths to maximize the amount
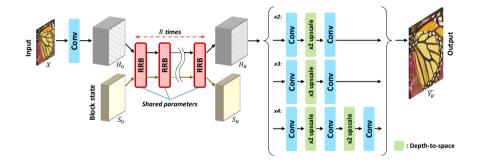
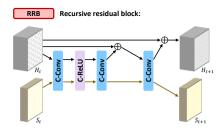**Fig. 2.** Overall structure of the proposed BSRN model.

Fig. 3. Structure of a recursive residual block (RRB).

of extracted information from the given low-resolution image. Along with the recursive super-resolution methods, the performance of these methods is also compared with that of our proposed method in Section 4.5.

We observe the following three common techniques from the previous work. First, increasing the spatial resolution at the latter stage can reduce the computational complexity than upscaling at the initial stage [3,6,12]. Second, employing multiple residual connections is beneficial to obtain better upscaled images [15,24]. Third, obtaining multiple upscaled images from the same super-resolution model and combining them into one provides better quality than acquiring a single image directly [14,19,25]. Along with the newly introduced block state-based architecture, our proposed method is built with considering the aforementioned empirical knowledge.

3   Proposed Method

In this section, we present how our super-resolution model works in detail. As similar to the existing super-resolution methods, our BSRN model can be divided into three parts: initial feature extraction, feature processing in a recursive manner, and upscaling. Fig. 2 shows the overall structure of our method. As

shown in the figure, the main objective of the super-resolution task is to obtain an image $\widehat{Y}$, which is upscaled from a given low-resolution image X, where we want $\widehat{Y}$ to be the same as the ground-truth image Y. Briefly, the initial features are extracted from the given input image. Then, the extracted features are further processed via a recursive residual block (RRB, Fig. 3), which is employed multiple times with the same parameters. The final image is obtained from the upscaling module.

3.1 Initial feature extractor

The BSRN model takes a low-resolution input image $X \in \mathbb{R}^{w \times h \times 3}$ consisting of three channels of the RGB color space, where $w \times h$ is the resolution of the image. Before we recursively process it, a convolutional layer extracts the initial features of the image, which can be represented as

$$H_0 = W_I * X + b_I \tag{1}$$

where $W_I \in \mathbb{R}^{3 \times 3 \times 3 \times c}$ and $b_I \in \mathbb{R}^c$ are the weight and bias matrices, respectively, and the operator $*$ denotes the convolution operation. A variable c determines the number of convolutional channels, thus the last dimension of H_0 is c.

3.2 Recursive residual block

Starting from the initial features H_0, our model performs the recursive operations in the shared part named "recursive residual block (RRB)," which is shown in Fig. 3. The RRB takes two matrices as inputs at a given iteration t: the feature matrix H_t that has been processed at previous iterations from the original input image and an additional matrix $S_t \in \mathbb{R}^{w \times h \times s}$ called "block state," where s determines the feature dimension of S_t. As shown in Fig. 2, the block state matrix is not derived from the input image features. Instead, the initial block state matrix S_0 is initialized by zero values. Note that S_t and H_t have the same spatial dimension but different feature dimensions.

A RRB consists of three concatenated convolution (C-Conv) layers and one concatenated rectified linear unit (C-ReLU) layer. A C-Conv layer first concatenates two input matrices along the last dimension, performs a convolutional operation, and splits the result into two output matrices having the sizes of the input matrices. In other words, when H_t and S_t are given, a C-Conv layer concatenates them (i.e., $[H_t, S_t]$), applies convolution as

$$[H_t', S_t'] = W_C * [H_t, S_t] + b_C \tag{2}$$

and splits them into $H_t' \in \mathbb{R}^{w \times h \times c}$ and $S_t' \in \mathbb{R}^{w \times h \times s}$, where $W_C \in \mathbb{R}^{3 \times 3 \times (c+s) \times (c+s)}$ and $b_C \in \mathbb{R}^{(c+s)}$ are the weight and bias matrices, respectively. A C-ReLU layer performs element-wise ReLU operations for the two inputs. In addition, two residual connections are involved for better performance as in the previous work [13,15]. After processing H_t and S_t with three C-Conv layers, one C-ReLU layer,

and two residual connections for the H_t part, the RRB outputs H_{t+1} and S_{t+1}, which then serve as the inputs of the same RRB for the next recursion. This recursive process is performed R times, which produces H_R and S_R.

There are two ways of configuring the BSRN model to get better performance: increasing the number of convolutional channels c and increasing the feature dimension of the block state s. When c increases, the number of model parameters increases across all parts of the model, including the initial feature extraction, RRB, and upscaling parts. On the other hand, increased s affects the number of model parameters only in the RRB part because the block state is involved only in RRB. Therefore, employing the block state is more beneficial to make the model compact than using a larger number of the convolutional channels.

In addition, because the block state can serve as a "memory," the RRB can keep track of the status of the current image features over the recursive operations. When the block state does not exist, it is hard to track the current status because it has to be latently written on the image features (i.e., H_t). It may lead to quality degradation of upscaled images, since both the image features and the current status are inputted to the upscaling part. We investigate the effectiveness of employing the block state in Section 4.3.

3.3   Upscaling

Finally, the BSRN model upscales the processed feature matrix H_R to generate an upscaled image $\widehat{Y_R}$. In particular, we use the depth-to-space operation as in the previous super-resolution models [3,6], which is also known as sub-pixel convolution [23]. For instance, in the upscaling part by a factor of 2, the first convolutional layer outputs the processed matrix having a size of $w \times h \times 4c$, the depth-to-space operator modifies the shape of the matrix to $2w \times 2h \times c$, and the last convolutional layer outputs the final upscaled image having a shape of $2w \times 2h \times 3$. Note that the block state S_t is not used in the upscaling part.

Our model can generate upscaled images not only from the final processed feature matrix H_R but also from the intermediate feature matrices H_t, $\forall t \in \{1, ..., R-1\}$. Therefore, with our model, it is possible to generate the upscaled images in a progressive manner. In addition, it is known that combining multiple outputs can improve the quality of the super-resolved images [9,14]. Thus, we adopt a similar approach to obtain the final upscaled image $\widehat{Y}$ by combining the intermediate outputs via the weighted sum as:

$$\widehat{Y} = \frac{\sum_{t=1}^{R/r} 2^{(rt-1)} \widehat{Y_{rt}}}{\sum_{t=1}^{R/r} 2^{(rt-1)}} \tag{3}$$

where r is a so-called "frequency control variable," which will be explained later. The term $2^{(rt-1)}$ controls the amount of contribution of each intermediate output, where the later outputs contribute to $\widehat{Y}$ more than the earlier outputs. This facilitates our model to generate intermediate upscaled images, which have progressively improved quality.

The variable r in (3) controls the frequency of the progressive upscaling. For example, when $R = 16$ and $r = 4$, $\widehat{Y}$ is obtained from the weighted sum of $\widehat{Y_4}$, $\widehat{Y_8}$, $\widehat{Y_{12}}$, and $\widehat{Y_{16}}$. Since a larger value of r reduces the number of times to employ the upscaling part, it is beneficial to reduce the processing time for generating the final super-resolved image. We discuss the influence of changing r in Section 4.4.

3.4   Loss function

The loss function of our model is calculated from the weighted sum of the pixel-by-pixel L1 loss, i.e.,

$$L(\widehat{Y}, Y) = \frac{1}{w' \times h'} \sum_{x=1}^{w'} \sum_{y=1}^{h'} \left| \widehat{Y}(x, y) - Y(x, y) \right| \tag{4}$$

where $w' \times h'$ is the spatial resolution of $\widehat{Y}$ and Y, and $\widehat{Y}(x, y)$ and $Y(x, y)$ are the pixel values at (x, y) of the upscaled and ground-truth images, respectively.

4   Experiments

We conduct three experiments to investigate the advantages of the BSRN model. First, we examine the effectiveness of employing the block state. Second, we explore the role of the frequency control variable r. Finally, we compare our models with the other state-of-the-art methods.

4.1   Dataset and evaluation metrics

We employ the DIV2K dataset [2] for training the BSRN models, which is widely used for training the recent super-resolution models [3,15]. For evaluating the performance of our models, we use four benchmark datasets, including Set5 [5], Set14 [29], BSD100 [20], and Urban100 [11].

We employ peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [27] for measuring quality of the upscaled images. As in the previous work [3,12], both metrics are calculated on the Y channel of the YCbCr channels converted from the RGB channels.

4.2   Training details

We build both single-scale ($\times 4$) and multi-scale ($\times 2$, $\times 3$, and $\times 4$) BSRN models. The single-scale models are used to find out the benefits of the block state and frequency control variable, and the multi-scale model is used to evaluate the performance of our model in comparison to the other super-resolution methods across different scales. The number of the recursive operations R and the frequency control variable r are set to 16 and 1, respectively.
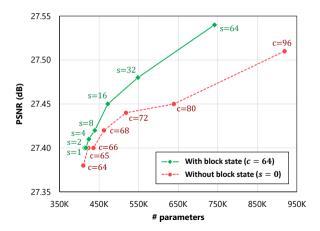
Fig. 4. Number of parameters and PSNR values of the ×4-scale BSRN models with and without the block state for the BSD100 dataset [20].

We implement the training and evaluation code of the BSRN model on the TensorFlow framework [1][1]. For each training step, eight image patches are randomly cropped from the training images. A cropping size of 32×32 pixels is used for training the single-scale BSRN model and 48×48 pixels is used for the multi-scale BSRN model. For data augmentation, the image patches are then randomly flipped and rotated. For the multi-scale BSRN model, one of the upscaling paths (i.e., ×2, ×3, and ×4) is randomly selected for every training step. The super-resolved images are obtained from our model by feeding the image patches. Then, the loss is calculated using (4) and the Adam optimization method [16] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\hat{\epsilon} = 10^{-8}$ is used to update the model parameters. To prevent the vanishing or exploding gradients problem [4], we employ the L2 norm-based gradient clipping method, which clips each gradient so as to fit its L2 norm within $[-\theta, \theta]$. In this study, we set $\theta = 5$. The initial learning rate is set to 10^{-4} and reduced by a half at every 2×10^5 training steps. A total of 1.0×10^6 and 1.5×10^6 steps are executed for training the single-scale and multi-scale BSRN models, respectively.

4.3   Benefits of employing the block state

As explained in Section 3.2, the BSRN model can be trained with various numbers of the convolutional channels (i.e., c) and the block state channels (i.e., s). Here, we investigate the effectiveness of employing the block state by comparing the single-scale BSRN models having an upscaling factor of 4, which are trained with and without using the block state. For the models with the block state, the number of the convolutional channels c is fixed to 64 and the number of

---

[1] The code is available at `https://github.com/idearibosome/tf-bsrn-sr`.
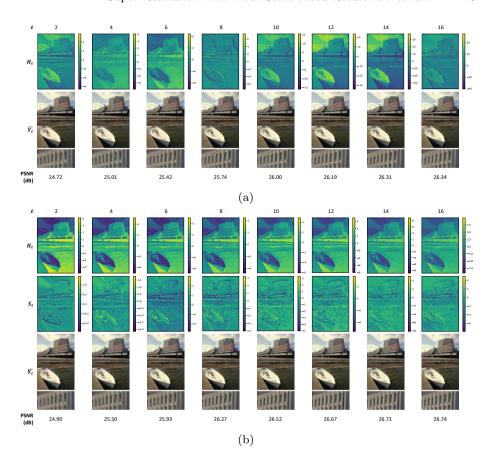
(a)

(b)

Fig. 5. Intermediate features H_t, block states S_t, upscaled images $\widehat{Y}_t$, and PSNR values of the BSRN models from an image of the BSD100 dataset [20]. Enlarged versions of the regions marked with red rectangles are also shown. (a) Without block state ($c = 64, s = 0$) (b) With block state ($c = 64, s = 64$)

the block state channels s is changed from 1 to 64. For the models without the block state, on the other hand, s is fixed to 0 and c is changed from 64 to 96. All models are tested with $r = 1$.

Fig. 4 compares the performance of the trained BSRN models in terms of the number of parameters and the PSNR values measured for the BSD100 dataset [20]. Overall, both the models with and without having the block states have a tendency to show better performance as the feature dimension (and consequently the number of parameters) increases. However, the BSRN models with the block state outperform the models without the block state, when the same numbers of parameters are used. This strongly supports that differentiating the place to store historical information from that for the image features helps to improve the quality of the upscaled images.

r	Processing time (s)	PSNR (dB)	SSIM
1	0.152	27.540	0.7341
2	0.094	27.540	0.7341
4	0.059	27.539	0.7341
8	0.047	27.538	0.7341
16	0.027	27.538	0.7341

Table 1. Performance comparison of the ×4-scale BSRN model tested with different values of r in terms of the average processing time to obtain $\widehat{Y}$, PSNR, and SSIM for the BSD100 dataset [20].

We further examine changes of the activation patterns of the BSRN models over the recursive iterations. Fig. 5 shows H_t, S_t, and $\widehat{Y}_t$ of the BSRN models trained with $c = 64, s = 0$ and $c = 64, s = 64$, where the corresponding PSNR values are also reported. The values of the intermediate features and block states are averaged along the last dimension. Both the models with and without the block state generate the upscaled images with gradually improved quality in terms of the PSNR values over the iteration t. However, the changes of the intermediate features are largely different. When the block state is not employed (Fig. 5 (a)), both the patterns of the activation and range of the values drastically change, even though the super-resolved images are not. This implies that the RRB of the model without the block state has difficulty in generating progressively improved features and highly relies on the latter part (i.e., upscaling part) to generate good quality of the upscaled images. On the other hand, employing the block state (Fig. 5 (b)) results in much more stable activations of H_t than the model without the block state. Instead, the block states S_t have major changes of the details, which provide historical information that can be used to produce gradually improved upscaled images over the iterations. This confirms that our model properly utilizes the block state along with the intermediate output features, which leads to better performance.

4.4   Role of frequency control variable (r)

Our model can be configured with the frequency of progressive outputs via r, along with the number of recursive iterations R. While R determines how many times the RRB is used to generate the final upscaled image, r determines how many intermediate images are obtained from the model to generate the final image (i.e., how many times the upscaling part is employed), which is R/r. Note that both R and r do not affect the number of model parameters.

In our proposed model, the upscaling part spends most of the computation time due to its increased number of the convolutional filters for the depth-to-space operation and increased spatial resolution after the depth-to-space operation. To verify this, we examine the BSRN model trained with $c = 64$ and $s = 64$ by testing with different values of r and compare their efficiency in terms of speed and quality of the upscaled images (i.e., PSNR and SSIM).

Scale	Method	# params	Set5 PSNR / SSIM	Set14 PSNR / SSIM	BSD100 PSNR / SSIM	Urban100 PSNR / SSIM
×2	VDSR [13]	666K	37.53 / 0.9587	33.03 / 0.9124	31.90 / 0.8960	30.76 / 0.9140
	DRCN [14]	1,774K	37.63 / 0.9588	33.04 / 0.9118	31.85 / 0.8942	30.75 / 0.9133
	LapSRN [18]	436K	37.52 / 0.959	33.08 / 0.913	31.80 / 0.895	30.41 / 0.910
	DRRN [24]	298K	37.74 / 0.9591	33.23 / 0.9136	32.05 / 0.8973	31.23 / 0.9188
	MemNet [25]	686K	37.78 / 0.9597	33.23 / 0.9142	32.08 / 0.8978	31.31 / 0.9195
	DSRN [9]	~1,200K	37.66 / 0.959	33.15 / 0.913	32.10 / 0.897	30.97 / 0.916
	IDN [12]	553K	37.83 / 0.9600	33.30 / 0.9148	32.08 / 0.8985	31.27 / 0.9196
	CARN [3]	964K	37.76 / 0.9590	33.52 / 0.9166	32.09 / 0.8978	31.51 / 0.9312
	BSRN (Ours)	594K	37.78 / 0.9591	33.43 / 0.9155	32.11 / 0.8983	31.92 / 0.9261
×3	VDSR [13]	666K	33.66 / 0.9213	29.77 / 0.8314	28.82 / 0.7976	27.14 / 0.8279
	DRCN [14]	1,774K	33.82 / 0.9226	29.76 / 0.8311	28.80 / 0.7963	27.15 / 0.8276
	DRRN [24]	298K	34.03 / 0.9244	29.96 / 0.8349	28.95 / 0.8004	27.53 / 0.8378
	MemNet [25]	686K	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	27.56 / 0.8376
	DSRN [9]	~1,200K	33.88 / 0.922	30.26 / 0.837	28.81 / 0.797	27.16 / 0.828
	IDN [12]	553K	34.11 / 0.9253	29.99 / 0.8354	28.95 / 0.8013	27.42 / 0.8359
	CARN [3]	1,149K	34.29 / 0.9255	30.29 / 0.8407	29.06 / 0.8034	27.38 / 0.8404
	BSRN (Ours)	779K	34.32 / 0.9255	30.25 / 0.8404	29.07 / 0.8039	28.04 / 0.8497
×4	VDSR [13]	666K	31.35 / 0.8838	28.01 / 0.7674	27.29 / 0.7251	25.18 / 0.7524
	DRCN [14]	1,774K	31.53 / 0.8854	28.02 / 0.7670	27.23 / 0.7233	25.14 / 0.7510
	LapSRN [18]	872K	31.54 / 0.885	28.19 / 0.772	27.32 / 0.728	25.21 / 0.756
	DRRN [24]	298K	31.68 / 0.8888	28.21 / 0.7720	27.38 / 0.7284	25.44 / 0.7638
	MemNet [25]	686K	31.74 / 0.8893	28.26 / 0.7723	27.40 / 0.7281	25.50 / 0.7630
	DSRN [9]	~1,200K	31.40 / 0.883	28.07 / 0.770	27.25 / 0.724	25.08 / 0.717
	IDN [12]	553K	31.82 / 0.8903	28.25 / 0.7730	27.41 / 0.7297	25.41 / 0.7632
	CARN [3]	1,112K	32.13 / 0.8937	28.60 / 0.7806	27.58 / 0.7349	26.07 / 0.7837
	BSRN (Ours)	742K	32.14 / 0.8937	28.56 / 0.7803	27.57 / 0.7353	26.03 / 0.7835

Table 2. Performance comparison of the state-of-the-art methods and our model evaluated on the Set5 [5], Set14 [29], BSD100 [20], and Urban100 [11] datasets. Red and blue colors indicate the best and second best performance, respectively.

Table 1 shows the average processing time spent on upscaling an image by a factor of 4, PSNR values, and SSIM values for the BSD100 dataset [20] for various values of r. The processing time is measured on a NVIDIA GeForce GTX 1080 GPU. As expected, the processing time largely decreases when r increases. For example, the BSRN model tested with $r = 16$ requires more than 5 times less processing time than the model tested with $r = 1$. Nevertheless, the PSNR value decreases by only 0.002 dB and SSIM value even remains the same. This confirms that increasing r significantly increases the processing speed with only negligible quality degradation. In addition, the experimental result implies that our proposed model has a capability of real-time processing. For example, when $r = 16$, our model can upscale more than 30 images per second, which is a common frame rate of videos.

4.5   Comparison with the other methods

Finally, we compare the performance of the multi-scale BSRN model with the other state-of-the-art super-resolution methods, including VDSR [13], DRCN [14], LapSRN [18], DRRN [24], MemNet [25], DSRN [9], IDN [12], and CARN
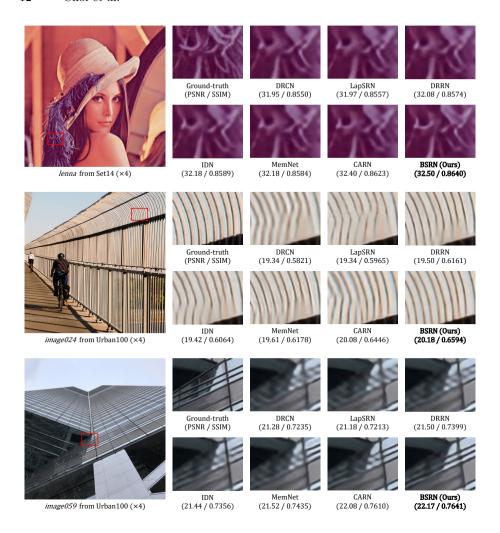
lenna from Set14 (×4)

| Ground-truth (PSNR / SSIM) | DRCN (31.95 / 0.8550) | LapSRN (31.97 / 0.8557) | DRRN (32.08 / 0.8574) |
| IDN (32.18 / 0.8589) | MemNet (32.18 / 0.8584) | CARN (32.40 / 0.8623) | **BSRN (Ours) (32.50 / 0.8640)** |

image024 from Urban100 (×4)

| Ground-truth (PSNR / SSIM) | DRCN (19.34 / 0.5821) | LapSRN (19.34 / 0.5965) | DRRN (19.50 / 0.6161) |
| IDN (19.42 / 0.6064) | MemNet (19.61 / 0.6178) | CARN (20.08 / 0.6446) | **BSRN (Ours) (20.18 / 0.6594)** |

image059 from Urban100 (×4)

| Ground-truth (PSNR / SSIM) | DRCN (21.28 / 0.7235) | LapSRN (21.18 / 0.7213) | DRRN (21.50 / 0.7399) |
| IDN (21.44 / 0.7356) | MemNet (21.52 / 0.7435) | CARN (22.08 / 0.7610) | **BSRN (Ours) (22.17 / 0.7641)** |

Fig. 6. Comparison of the upscaled images obtained by the BSRN model and the other state-of-the-art methods.

[3]. The DRCN, DRRN, MemNet, DSRN, and CARN models contain parameter-sharing parts. The VDSR, LapSRN, and IDN methods are also included in the comparison, since they have been recently proposed and have similar numbers of model parameters to ours.

Table 2 shows the performance of the state-of-the-art methods and ours in terms of the PSNR and SSIM values on the four benchmark datasets. The number of model parameters required to obtain the super-resolved image with the given upscaling factor for each method is also provided. First, the BSRN model outperforms the other methods that do not employ any recursive operations

or parameter-sharing, including VDSR, LapSRN, and IDN. For example, our method achieves a quality gain of 0.31 dB for a scale factor of 2 on the BSD100 dataset over the LapSRN model. It confirms that recursive processing helps to obtain better super-resolved images with keeping the number of model parameters small enough.

In addition, our model employs much less numbers of parameters than DRCN, DSRN, and CARN. For instance, the BSRN model uses up to 70% less numbers of model parameters than the DRCN model. Nevertheless, our proposed model outperforms DRCN and DSRN, and shows comparable performance to CARN. In particular, BSRN shows almost the same performance as CARN despite the smaller model size. This proves that the proposed method handles the image features better than the other state-of-the-art methods.

Fig. 6 provides a showcase of the images reconstructed by our proposed model and the other state-of-the-art methods. The figure shows that the BSRN model is highly reliable in recovering textures from the low-resolution images. For example, our method successfully upscales fine details of the structures in the Urban100 dataset, which results in clearer outputs, while the other methods produce highly blurred images or images containing large amounts of artifacts. This confirms that the BSRN model produces images having visually nice super-resolved images.

5   Conclusion

In this paper, we introduced the BSRN model, which employs a novel way of recursive operation using the block state, for the super-resolution tasks. We explained the benefits and efficiency of employing our model in terms of the number of model parameters, quality measures (i.e., PSNR and SSIM), and speed. In addition, comparison with the other state-of-the-art methods also showed that our method can generate better quality of the upscaled images than the others.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: TensorFlow: A system for large-scale machine learning. In: Proceedings of the USENIX Symposium on Operating Systems Design and Implementation. pp. 265–283 (2016)
2. Agustsson, E., Timofte, R.: NTIRE 2017 challenge on single image super-resolution: Dataset and study. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 126–135 (2017)
3. Ahn, N., Kang, B., Sohn, K.A.: Fast, accurate, and lightweight super-resolution with cascading residual network. In: Proceedings of the European Conference on Computer Vision. pp. 252–268 (2018)
4. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5(2), 157–166 (1994)
5. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: Proceedings of the British Machine Vision Conference. pp. 1–10 (2012)

6. Choi, J.S., Kim, M.: A deep convolutional neural network with selection units for super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 1150–1156 (2017)

7. Choi, J.H., Choi, M., Choi, M.S., Lee, J.S.: Impact of three-dimensional video scalability on multi-view activity recognition using deep learning. In: Proceedings of the Thematic Workshops of ACM Conference on Multimedia. pp. 135–143 (2017)

8. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: Proceedings of the European Conference on Computer Vision. pp. 184–199 (2014)

9. Han, W., Chang, S., Liu, D., Yu, M., Witbrock, M., Huang, T.S.: Image super-resolution via dual-state recurrent networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1654–1663 (2018)

10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)

11. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5197–5206 (2015)

12. Hui, Z., Wang, X., Gao, X.: Fast and accurate single image super-resolution via information distillation network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 723–731 (2018)

13. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1646–1654 (2016)

14. Kim, J., Lee, J.K., Lee, K.M.: Deeply-recursive convolutional network for image super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1637–1645 (2016)

15. Kim, J.H., Lee, J.S.: Deep residual network with enhanced upscaling module for super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 800–808 (2018)

16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations. pp. 1–13 (2015)

17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)

18. Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Deep Laplacian pyramid networks for fast and accurate super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 624–632 (2017)

19. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. In: Proccedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 136–144 (2017)

20. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 416–423 (2001)

21. Montufar, G.F., Pascanu, R., Cho, K., Bengio, Y.: On the number of linear regions of deep neural networks. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 2924–2932 (2014)

22. Salvador, J., Perez-Pellitero, E.: Naive Bayes super-resolution forest. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 325–333 (2015)

23. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1874–1883 (2016)
24. Tai, Y., Yang, J., Liu, X.: Image super-resolution via deep recursive residual network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3147–3155 (2017)
25. Tai, Y., Yang, J., Liu, X., Xu, C.: MemNet: A persistent memory network for image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4539–4547 (2017)
26. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3156–3164 (2015)
27. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004)
28. Yang, S., Liu, Z., Wang, M., Sun, F., Jiao, L.: Multitask dictionary learning and sparse representation based single-image super-resolution reconstruction. Neurocomputing **74**(17), 3193–3203 (2011)
29. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Proceedings of the International Conference on Curves and Surfaces. pp. 711–730 (2010)