LatticeNet: Towards Lightweight Image Super-Resolution with Lattice Block

Xiaotong Luo[1], Yuan Xie[2], Yulun Zhang[3], Yanyun Qu[1(✉)], Cuihua Li[1], and Yun Fu[3]

[1] School of Informatics, Xiamen University, Xiamen, China
luoxiaotong@stu.xmu.edu.cn, {yyqu,chli}@xmu.edu.cn
[2] School of Computer Science and Technology, East China Normal University, Shanghai, China
yxie@cs.ecnu.edu.cn
[3] Department of ECE, Northeastern University, Boston, USA
yulun100@gmail.com, yunfu@ece.neu.edu

Abstract. Deep neural networks with a massive number of layers have made a remarkable breakthrough on single image super-resolution (SR), but sacrifice computation complexity and memory storage. To address this problem, we focus on the lightweight models for fast and accurate image SR. Due to the frequent use of residual block (RB) in SR models, we pursue an economical structure to adaptively combine RBs. Drawing lessons from lattice filter bank, we design the lattice block (LB) in which two butterfly structures are applied to combine two RBs. LB has the potential of various linear combinations of two RBs. Each case of LB depends on the combination coefficients which are determined by the attention mechanism. LB favors the lightweight SR model with the reduction of about half amount of the parameters while keeping the similar SR performance. Moreover, we propose a lightweight SR model, LatticeNet, which uses series connection of LBs and the backward feature fusion. Extensive experiments demonstrate that our proposal can achieve superior accuracy on four available benchmark datasets against other state-of-the-art methods, while maintaining relatively low computation and memory requirements.

Keywords: Super-resolution · Lattice block · LatticeNet · Lightweight · Attention

1 Introduction

Single image super-resolution (SISR) pursues to recover a high-resolution (HR) image from its degraded low-resolution (LR) counterpart. The arise of convolutional neural networks, accompanying with the residual learning [12], have paved

---

X. Luo and Y. Xie—Equal contribution.

the way for the development of SISR. With the massive number of stacked residual blocks (RBs), the existing deep SR models [21,25,35] achieve great breakthrough in accuracy. However, they cannot be easily utilized to real applications for the high computational complexity and memory storage.

To reduce model parameters, most existing works still focus on the architecture design, such as pyramid network [20], the recursive operation with weight sharing [2,5], channel grouping [16,17] and neural architecture search [6]. As we know, RB as a basic unit is often utilized in many SR methods. Therefore, we aim to explore how to make a lightweight model in the view of RBs.
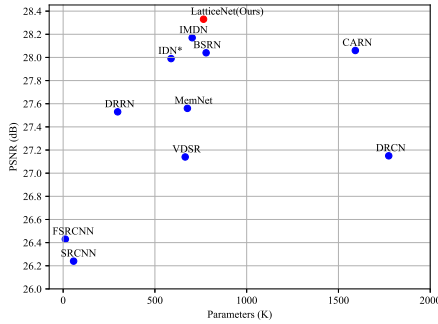

Fig. 1. Parameters and accuracy trade-off with the SOTA lightweight methods on Urban100 for $3\times$ SR. LatticeNet achieves superior performance with moderate size

Inspired by the lattice filter bank [22] which is the physical realization of Fast Fourier Transformation with the butterfly structure, we design the lattice block (LB) with two butterfly structures, each of which accompanies with a RB. LB is an economical structure with the potential of various linear combination patterns of two RBs, **which can help to expand representation space for achieving a more powerful network**. The series connection of two RBs is only a special case of LB. Moreover, in order to obtain the appropriate combination of RBs for SR, the connection weights of pair-wise RBs in LB, named combination coefficients, are learned with the attention mechanism for information reweighting rather than from scratch.

Based on LB, we build a lightweight SR network named LatticeNet. Compared with the recursive operation [5] and channel grouping [16], LB paves a new way to a better architectural design of RB combination. Besides, a feature fusion module is used to integrate multiple hierarchical features from different receptive fields in a backward concatenation strategy, which helps to mine feature correlations of intermediate layers. Unlike the latest work IMDN [16], which uses channel split similar to DenseNet for feature learning, LB is a more general unit and can be used in the previous SR models to replace RBs, which leads to a lightweight SR model. LatticeNet is superior to the state-of-the-art lightweight SR models as shown in Fig. 1. To summarize, the main contributions are:

– The lattice block (LB) is elaborately designed based on the lattice filter bank with two butterfly structure. With the help of this structure, the network representation capability can be significantly expanded through diverse combinational patterns of residual blocks (RBs).
– LB is adaptively combined by the learnable combination coefficients of the RBs with the attention mechanism, which upweights the important channels of feature maps to obtain the better SR performance.
– LB favors lightweight model design. Based on the novel block, we build a lightweight SR network dubbed LatticeNet with the backward fusion strategy for extracting hierarchical contextual information.
– LB leads to the reduction of parameters by about half in the baseline SR models while keeping the similar SR performance if RBs are replaced by LBs. LatticeNet achieves the superior performance on several SR benchmark datasets against the state-of-the-art methods while maintaining relatively lower model size and computational complexity.

2   Related Work

2.1   Deep SR Models

Numerous deep SR models have been proposed and achieved promising performance. SRCNN [8] is the preliminary work for applying a three-layer convolutional neural network to the SR task. VDSR [18] and MemNet [30] employ skip connections for learning the residual information. RCAN [35] proposes a deep residual network for SR with the residual-in-residual structure embedded with the channel-wise attention mechanism. SAN [7] proposes a second-order attention SR network for the purpose of powerful feature expression and correlation learning. SRFBN [24] proposes a feedback mechanism to address the feedback connections and generate effective high-level feature representations. CFSNet [31] adaptively learns the coupling coefficients from different layers and feature channels for finer controlling of the recovered image. EBRN [33] proposes an incremental recovering process for texture SR. Although these deep SR models can make significant performance quantitatively and qualitatively, they are highly cost in memory storage and computational complexity.

2.2   Lightweight SR Models

Lightweight models have attracted widespread attention for saving computing resources. They can be approximately divided into three classes: the architectural design related methods [17,29], the knowledge distillation based methods [11], and the neural architecture search based methods [6]. The first kind of lightweight SR methods mainly focus on recursive operation and channel splitting. DRCN [19] first applies the recursive neural network to SR. DRRN [29] adopts a deeper network with the recursive operation. CARN [2] utilizes a recursive cascading mechanism for learning multi-level feature representations. BSRN

[5] employs a block state-based recursive network and performs well in quality measures and speed. IDN [17] utilizes group convolution and combines the local long and short-path features effectively. IMDN [16] introduces information multi-distillation blocks to enlarge receptive field for extracting hierarchical features. Recently, knowledge distillation is used to learn a lightweight student SR network from a deep and powerful teacher network in [11]. Besides, FALSR [6] applies neural architecture search to SR and performs excellently. Although the researches of lightweight SR models have made great progress, it is still in the primary stage and more discussions are required.

2.3   Attention Mechanism

Nowadays, the attention mechanism emerges in numerous deep neural networks. SENet [14] firstly proposes a lightweight module to treat channel-wise features differently according to the respective weight responses. The attention module is introduced for low-level image restoration in [23,35]. It has also been implemented on many other tasks. DANet [10] introduces two parallel attention modules to model the semantic interdependencies in spatial and channel dimensions respectively for scene segmentation. CBAM [32] also infers attention maps along two separate dimensions including channel and spatial. Due to the effectiveness of attention models, we also embed the attention mechanism into the lattice block to combine the RBs adaptively.

3   Proposed Method

In this section, we first present our lattice block based on the lattice filter. The lattice block includes two components: the topological structure and the connection weights. The former is a butterfly structure, and the latter is computed by using the attention mechanism. Then, we describe the overall network architecture. Next, the loss function is defined to optimize the model. Finally, we discuss the differences between the proposed method and its related works.

3.1   From Lattice Filter to Lattice Block

Suppose there are two signals x and y, the linear combination O between them has the following types: 1) O = ax + y; 2) O = x + by; 3) O = ax + by, where a, b denote different weights. Generally, the three formulations are equivalent. The first two types are similar to the form of the identity mapping residual learning [13]. Therefore, we consider to employ such combination.

Lattice Filter. The structure of lattice filter is a variant of the decimation-in-time butterfly operation of FFT, which decomposes the input signal to multi-order representations. Figure 2(a) shows the basic unit of the standard lattice structure for a 2-channel filter bank, and the relationship between the input and output is formalized in Eq. (1) and Eq. (2). Here, z represents the variable of the

z-transformation. z^{-1} corresponds to the delay of one unit of the signal in the time domain. The high-order components $P_i(z)$ and $Q_i(z)$ can be synthesized from low-order inputs $P_{i-1}(z)$ and $Q_{i-1}(z)$ by a crossed way. a_i denotes the coefficient which defines the combination between two components. It can achieve high-speed parallel processing of FFT for the modular structure [22].

$$P_i(z) = P_{i-1}(z) + a_i z^{-1} Q_{i-1}(z), \tag{1}$$

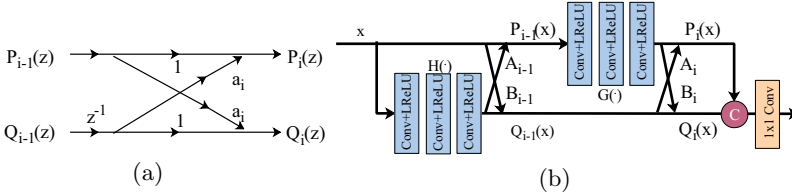$$Q_i(z) = a_i P_{i-1}(z) + z^{-1} Q_{i-1}(z). \tag{2}$$

Fig. 2. (a) The basic unit of a standard lattice structure for a 2-channel filter bank. (b) The structure of proposed lattice block

Lattice Block. Inspired by the lattice filter bank, we design the lattice block (LB) which contains two such butterfly structures with a RB per butterfly structure, as shown in Fig. 2(b). The butterfly structure favors multiple combination patterns of RBs. Thus, LB is an economical structure with the potential of various combinations of two RBs. In detail, a collection of features maps $\mathcal{X}$ is fed into in the lower branch which contains three convolutional layers with a Leaky Rectified Linear Unit (LReLU) activation function per layer. The nonlinear function implicitly induced by all these operators is denoted as $H(\cdot)$. Then, the first combination between $\mathcal{X}$ and $H(\mathcal{X})$ can be formulated as

$$P_{i-1}(\mathcal{X}) = \mathcal{X} + A_{i-1} H(\mathcal{X}), \tag{3}$$

$$Q_{i-1}(\mathcal{X}) = B_{i-1} \mathcal{X} + H(\mathcal{X}), \tag{4}$$

where A_{i-1} and B_{i-1} are two vectors of combination coefficients, which are adaptively computed according to the responses of features (i.e., $H(\mathcal{X})$, $\mathcal{X}$) for information reweighting. Considering a convolutional layer contains multiple channels, each channel can be viewed as a signal. Therefore, the combination coefficients A_{i-1} and B_{i-1} are vectors, whose length is equal to the number of feature maps.

Next, $P_{i-1}(\mathcal{X})$ is fed into in the upper branch to go through the same convolutional structure as $H(\cdot)$ defined by $G(\cdot)$. Then, the second combination of $P_i(\mathcal{X})$ and $Q_i(\mathcal{X})$ can be formulated as

$$P_i(\mathcal{X}) = G(P_{i-1}(\mathcal{X})) + A_i Q_{i-1}(\mathcal{X}), \tag{5}$$

$$Q_i(\mathcal{X}) = B_i G(P_{i-1}(\mathcal{X})) + Q_{i-1}(\mathcal{X}). \tag{6}$$

where A_i and B_i are also the combination coefficients corresponding to $Q_{i-1}(\mathcal{X})$ and $G(P_{i-1}(\mathcal{X}))$. After that, $P_i(\mathcal{X})$ and $Q_i(\mathcal{X})$ are concatenated in channels and followed by a 1×1 convolution for channel alignment.

Potential Combinations in LB. Here, we mainly analyse the multiple combinations of RBs in LB. Given input feature maps $\mathcal{X}$, the output $\mathcal{Y}$ before the 1×1 convolution of LB is denoted as

$$\mathcal{Y} = concat(A_i(H(\mathcal{X}) + B_{i-1}\mathcal{X}) + G(\mathcal{X} + A_{i-1}H(\mathcal{X})),$$
$$H(\mathcal{X}) + B_{i-1}\mathcal{X} + B_i(G(\mathcal{X} + A_{i-1}H(\mathcal{X})))). \tag{7}$$
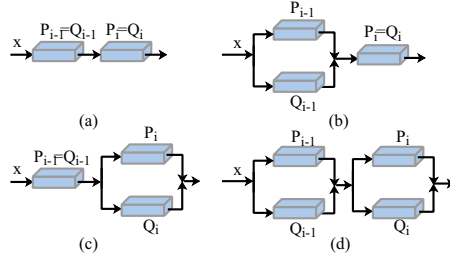

Fig. 3. Examples of various combination patterns of an LB, where the blue block denotes a RB. (a) Two sequential RBs. (b) Two parallel RBs followed by a RB. (c) A RB followed by two parallel RBs. (d) Two parallel RBs followed by two parallel RBs

In a unrolled view, an LB contains the potential of multiple combination patterns corresponding to different combination coefficients. Note that $\mathbf{1}$ and $\mathbf{0}$ denote the vectors whose elements are 1 and 0, respectively. Assuming the following special cases:

(1) $\boldsymbol{A_{i-1}} = \boldsymbol{B_{i-1}} = \boldsymbol{A_i} = \boldsymbol{B_i} = \mathbf{1}$. $P_{i-1} = Q_{i-1} = \mathcal{X} + H(\mathcal{X})$, $P_i = Q_i = P_{i-1} + G(P_{i-1})$. The two branches are identical, so that the LB can be simplified to series connection of two RBs as shown in Fig. 3(a).

(2) $\boldsymbol{A_{i-1}} \neq \boldsymbol{B_{i-1}}, \boldsymbol{A_i} = \boldsymbol{B_i} = \mathbf{1}$. $P_{i-1} = \mathcal{X} + A_{i-1}H(\mathcal{X})$, $Q_{i-1} = H(\mathcal{X}) + B_{i-1}\mathcal{X}$, and P_{i-1} is not equal to Q_{i-1}. $P_i = G(P_{i-1}) + Q_{i-1}$, $Q_i = Q_{i-1} + G(P_{i-1})$. Note that P_i is equal to Q_i. As shown in Fig. 3(b), the LB is degraded to two parallel scaling RBs with weights sharing followed by a RB.

(3) $\boldsymbol{A_{i-1}} = \boldsymbol{B_{i-1}} = \mathbf{1}, \boldsymbol{A_i} \neq \boldsymbol{B_i}$. $P_{i-1} = Q_{i-1} = \mathcal{X} + H(\mathcal{X})$, $P_i = G(P_{i-1}) + A_iQ_{i-1}$, $Q_i = Q_{i-1} + B_iG(P_{i-1})$. Note that P_i is not equal to Q_i. As shown in Fig. 3(c), LB is degraded to a RB followed by two parallel scaling RBs with weights sharing.

(4) $\boldsymbol{A_{i-1}} \neq \boldsymbol{B_{i-1}}, \boldsymbol{A_i} \neq \boldsymbol{B_i}$. $P_{i-1} \neq Q_{i-1}$, $P_i \neq Q_i$. As shown in Fig. 3(d), the LB is equivalent to two parallel scaling RBs with weights sharing followed by two parallel scaling RBs with weights sharing.

There still exist other combination patterns. For example, when all the coefficients are equal to **0**, the LB is equivalent to two parallel stacks of convolution layers. These special cases can be approximately achieved by normalized coefficients. Therefore, the proposed LB can be viewed as diverse combination patterns of RBs to expand the representation space with the learnable combination coefficients. The diverse structures of LB favor lightweight model design.

Combination Coefficient Learning. The vectors A_i and B_i of combination coefficients actually play the role of the connection weights in LB, as shown in Fig. 2(b). As mentioned above, the special vectors of combination coefficients are related to the special cases of LB in the unrolled view. Rather than brute-force searching all the potential structures in LB, we adopt the attention mechanism to compute the combination coefficients. Inspired by the success of SENet [14] and IMDN [16], we utilize two statistics to compute the combination coefficients: the mean and the standard deviation of a feature map.

As Fig. 4 shows, given a group of feature maps, in the upper branch, we get the mean value of each feature map by global average pooling, and in the lower branch, we compute the standard deviation of each feature map. After that, the statistic vector in each branch is passed to two 1×1 convolution layers, each of which is followed by the ReLU activation layer and Sigmoid activation layer, respectively. Finally, the output vectors of two branches are averaged as the combination coefficients.
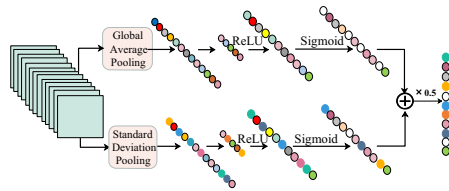

Fig. 4. The combination coefficient learning. Mean and standard deviation of each channel are used to describe the statistics of input features

3.2   Network Architecture

Based on the lattice block, we propose a lightweight SR network dubbed LatticeNet, which contains four components: the shallow feature extraction, multiple cascaded lattice blocks (LB), the backward fusion module (BFM), and the upsampling module as shown in Fig. 5. Here, the input LR images and the output SR images are denoted as X and Y, respectively.

Firstly, we obtain the shallow features F_0 by applying two cascaded 3×3 convolutional layers without activation to the LR input X:

$$F_0 = R_0(X), \tag{8}$$

where $R_0(\cdot)$ denotes the shallow convolution operation.

Several LBs are followed for deep feature interaction and mapping, which can be formulated as

$$F_k = R_k(F_{k-1}), k = 1, \ldots, M, \tag{9}$$

where $R_k(\cdot)$ indicates mapping function of the k-th LB, F_{k-1} represents the features from the previous adjacent LB, and M is the total number of LBs.

BFM is used to integrate features from all LBs for extracting hierarchical frequency information:

$$F_f = R_f(F_1, F_2, ..., F_M), \tag{10}$$

where $R_f(\cdot)$, F_f are the backward feature fusion function and the fused features, respectively. Finally, the upsampling layer with sub-pixel convolution [27] is utilized for generating the SR image Y:

$$Y = R_{up}(F_f + F_0), \tag{11}$$

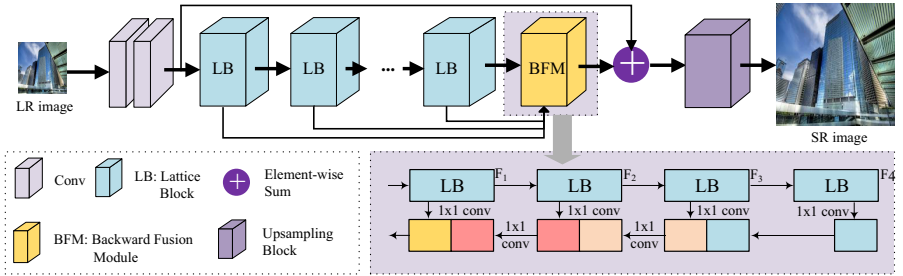where $R_{up}(\cdot)$ denotes the upsampling function.

Fig. 5. The overall network architecture of the proposed LatticeNet

3.3   Backward Fusion Module

The hierarchical information is very important for SR. Thus, we fuse features from multiple layers to obtain more contextual information. In this paper, we adopt a backward sequential concatenation strategy for feature fusion of different receptive fields like [28]. The core operation of our feature fusion module is the 1×1 convolution followed by the ReLU activation function (omitted in Fig. 5) for reducing feature dimension by half. Here, we denote the output of the i-th LB as F_i. As shown in Fig. 5, there are four LBs. The features of each LB are firstly convolved by 1×1 kernel which results in dimension reduction by half. Let F_i ($i = 1, 2, 3, 4$) denote the obtained features with the index increasing from the left to right in Fig. 5. The fusion operation is formulated as

$$H_i = \begin{cases} F_i, i = 4, \\ Conv(Concat(F_i, H_{i+1})), i = 3, 2, 1 \end{cases} \tag{12}$$

where $Concat(F_i, H_{i+1})$ denotes the operation that concatenates F_i and H_{i+1} and $Conv()$ denotes 1×1 convolution. In detail, the last two adjacent groups of feature maps F_i and F_{i-1} are concatenated and then convolved with 1×1 kernel. After that, the obtained feature group H_i is concatenated with F_{i-1}, which follows a 1×1 convolution, and so forth. The final fused features H_1 adding the shallow features F_0 are propagated to the upsampling layer to generate SR images. By such a backward sequential concatenation way, the feature fusion module can integrate the features from all the LBs, which helps to extract more hierarchical contextual information.

3.4   Loss Function

Mean Absolute Error (MAE) and Mean Squared Error (MSE) loss are the most frequently used loss functions for low-level image restoration tasks. Here, we only adopt MAE loss for measuring the differences between the SR images and the ground truth. Specifically, the loss function is

$$L = \frac{1}{N} \sum_{i=1}^{N} \left\| x_i^{gt} - R(x_i^{lr}) \right\|_1, \tag{13}$$

where x_i^{lr}, x_i^{gt} denote the i-th LR patch and the corresponding ground truth. N is the total number of training samples. $R(\cdot)$ represents the output of LatticeNet.

3.5   Discussions

In this section, we discuss the differences between the proposed model and the related works, e.g., SRResNet [21] and RDN [36].

LatticeNet vs. Residual Network. Residual blocks are popularly used in for image restoration. SRResNet [21] and RCAN are two typical residual models for SISR. The former adopts cascaded RBs as feature mapping and the later proposes a residual-in-residual structure with channel attention. In LatticeNet, the structure of two cascaded RBs is just a special case of LB. The proposed block can represent more diverse structure patterns and can be embedded in any SR networks with RBs as basic block.

LatticeNet vs. RDN. In RDN [36], the feature fusion adopts 1×1 convolution for concatenating previous features directly. In LatticeNet, BFM is used to combine those features by gradual concatenation approach.

The proposed lattice structure is not just another kind of residual and dense connection. **The insight behind it is a kind of network architecture search for finding reasonable combinations of RBs to make a lightweight SR model.** Rather than exhausted search, LB is a butterfly structure borrowed from signal processing and never investigated in SR before.

4   Experiments

4.1   Datasets

The model is trained with a high-quality dataset DIV2K [1], which is widely used for image SR task. It includes 800 training images and 100 validation images with rich textures. The LR images are obtained in the same way as [2,35]. Besides, we evaluate our model on four public SR benchmark datasets: Set5 [4], Set14 [34], B100 [3], and Urban100 [15].

4.2   Implementation Details

During training, 48×48 RGB image patches are input to the network. The training data is augmented by random flipping and rotation. We use Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ to train the model. The mini-batch size is 16. The learning rate is initialized as $2e - 4$ and reduced by half per 200 epochs for 1000 epochs totally. The kernel size of all convolutional layers is set to 3×3 except for 1×1 convolutional layer. The number of convolutional kernels for each non-linear function in the LB is set to 48, 48 and 64, respectively. We use PyTorch to implement our model with a GTX 1080 GPU. It takes about one day for training LatticeNet.

We use objective criteria, i.e., peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) to evaluate our model performance. The two metrics are both calculated on the Y channel of the YCbCr space as adopted in the previous work [35]. Besides, we use Mult-Adds to evaluate the computational complexity of a model, which denotes the number of composite multiply-accumulate operations for a single image. Similar to [2], we also assume the size of a query image is 1280×720 to calculate Mult-Adds.

4.3   The Contribution of Lattice Block for Lightweight

As mentioned in Sect. 3.5, we embed LB in large models for validating its effectiveness. Every four consecutive RBs are replaced with an LB in SRResNet [21] and RCAN [35] (called SRResNet_LB and RCAN_LB). For a fair comparison, the combination coefficients are calculated only by global average pooling for the same configuration with RCAN.

As Table 1 shows, SRResNet_LB achieves the comparable performance on all the datasets against SRResNet* with 0.96M parameters and the PSNR value even outperforms SRResNet* by +0.02 dB on Set5. Moreover, RCAN_LB also obtains the similar PSNR values against RCAN with only 8.6M parameters, which has approximately half amount of parameters than RCAN.

4.4   Ablation Analysis

In this subsection, we discuss LatticeNet and first analyse the effects of lattice block (LB), backward fusion module (BFM) and combination coefficients (CC).

Table 1. Quantitative evaluation in PSNR on four benchmark datasets for 4× SR. SRResNet* is the reimplemented results with DIV2K dataset. The best PSNR (dB) values are bold

Models	Set5	Set14	B100	Urban100	Params
SRResNet	31.92	28.39	27.52	-	1.52M
SRResNet*	32.18	28.64	27.59	26.19	1.52M
SRResNet_LB	**32.20**	28.62	27.56	26.14	**0.96M**
RCAN	**32.63**	28.87	**27.77**	**26.82**	15.6M
RCAN_LB	32.59	**28.88**	27.75	26.78	**8.6M**

The baseline only contains four basic blocks, each of which only contains two cascaded RBs with three convolution layers (the number of kernels is 48, 48, 64, respectively) without BFM, and other components are similar to LatticeNet. Then, we discuss the performance of several combination patterns of pair-wise RBs. Finally, we give how the number of LBs affects SR performance and the related comparison in running time.

Lattice Block (LB). To demonstrate the effect of LB, we replace the basic block in the baseline with LB. As Table 2 shows, the PSNR values of the baseline on Set5 and Set14 for 2× SR are the lowest. After adding the butterfly structure of LB, the PSNR values are increased by +0.12 dB, +0.16 dB on Set5 and Set14 with only ∼50K parameters increased as the column "1st" illustrates. Besides, the baseline with LB gains 0.12 dB in Set5 while the baseline with BFM only gains 0.04 dB, which shows that LB contributes more to the SR performance than BFM. It indicates that LB improves the expression of the network through the butterfly combination patterns.

Table 2. Ablation studies of the effects of lattice block (LB), backward fusion module (BFM). We give the best PSNR (dB) values on Set5 and Set14 for 2× SR in 500 epochs

Options	Baseline	1st	2nd	3rd	4th
LB	×	✓	×	×	✓
BFM	×	×	1×1	✓	✓
Params	692K	743K	709K	704K	756K
PSNR on Set5	37.84	37.96	37.85	37.88	**38.01**
PSNR on Set14	33.43	33.59	33.45	33.47	**33.65**

Backward Fusion Module (BFM). To evaluate the effect of the BFM, we add BFM to the baseline network. As the "3rd" column shows in Table 2, the PSNR values with BFM are both increased by +0.04 dB on Set5 and Set14 with

only ∼12K parameters increased upon the baseline. Besides, we also compare BFM with 1×1 convolution fusion that is used in RDN [36] for fusing all features from all LBs directly. It is observed that BFM is superior to the fusion strategy using 1×1 convolution with less parameters. Moreover, BFM is similar to FPN [26] which is proven to be effective in object detection. The baseline combined with BFM and LB achieves better SR performance whose PSNR is increased from 37.84 dB to 38.01 dB on Set5. The experiments show the effectiveness of the lightweight fusion module.

Combination Coefficients (CC). To fully employ the statistical characteristics of data, we compute combination coefficients by integrating mean pooling (MP) and standard deviation pooling (SDP). The experimental results show that a little gain can be achieved on four benchmark datasets if we put them together as illustrated in Table 3. Note that the combination coefficients in Table 2 adopt the mean and standard deviation. It shows that the attention mechanism for low-level vision tasks can be further investigated.

Table 3. Analysis of the effect of different combination coefficients (CC). We give the best PSNR (dB) values on four benchmark datasets for 2× SR in 500 epochs

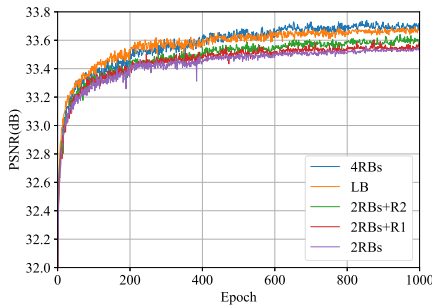| CC	Set5	Set14	B100	Urban100	Params
MP	37.985	33.621	32.099	32.086	747K
SDP	37.988	33.628	32.106	32.099	747K
MP & SDP	**38.012**	**33.647**	**32.124**	**32.145**	756K

Fig. 6. Comparisons of four different RB structures: 2RBs, 2RBs+R1, 2RBs+R2, 4RBs with LB on Set14 for 2× SR

Comparisons of Different RB Structures. Considering that the reason for the PSNR improvement of LB may be the increase of parameters, we compare several RB structures with LB for the amount of computation and parameters. We place these structures as basic block in LatticeNet, which are defined as:

Table 4. Trade-off between parameters, accuracy and speed of LatticeNet for 4× SR. We give the best PSNR values and execution time on Set5 and Set14 in 500 epochs

Num	Params	Set5	Set14
		PSNR(dB)/Time(s)	PSNR(dB)/Time(s)
2	438K	32.03/0.0328	28.50/0.0159
4	777K	32.18/0.0355	28.61/0.0203
8	1455K	32.29/0.0471	28.68/0.0301
10	1794K	32.35/ 0.0515	28.74/0.0343

(1) 2RBs: two cascaded RBs, which has slightly fewer parameters and computation amount than an LB structure.
(2) 2RBs+R1: two cascaded RBs, each of which goes through a recursive operation. It has slightly fewer parameters and approximately twice computation amount than an LB structure.
(3) 2RBs+R2: two cascaded RBs with once recursive operation, which has the same parameters and computation amount with 2RBs+R1.
(4) 4RBs: four cascaded RBs, which has approximately twice parameters and computation amount than an LB.

The quantitative evaluation of these structures in PSNR for 2× SR is illustrated in Fig. 6. The experimental results show an LB is not only superior to 2RBs, but also better than 2RBs+R1 and 2RBs+R2. Moreover, LB can even obtain comparable performance with 4RBs, which has near twice parameters than an LB. Therefore, it can demonstrate that the performance improvement really comes from the lattice structure and LB favors lightweight model design.

Number of Lattice Blocks. For better balancing the model size, performance and execution time, we compare the proposed model with different number of LBs, i.e., 2, 4, 8, 10. As shown in Table 4, with the number of LBs increasing, the SR performance can be improved, accompanying with parameters and execution time rising up. Therefore, we use 4 LBs in our experiments. Besides, the average running time comparison in our experimental environment between IMDN [16] and LatticeNet is shown in Table 5, where the testing code is given by IMDN. It shows that the differences of running time are negligible although the parameter of LatticeNet is slightly larger than IMDN.

Table 5. Average inference time (ms) comparisons between IMDN and LatticeNet on four benchmark datasets for 4× SR

Model	Set5	Set14	B100	Urban100	Depth
IMDN	**34.1**	22.1	**9.3**	39.9	34
LatticeNet	35.5	**20.3**	11.1	**12.0**	32

Table 6. Average PSNR/SSIM for 2×, 3×, 4× SR. The best results are highlighted in red color and the second best is in blue

Scale	Method	Params	Mult-Adds	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM
2×	Bicubic	-	-	33.66/0.9299	30.24/0.8688	29.56/0.8431	26.88/0.8403
	SRCNN [8]	57K	52.7G	36.66/0.9542	32.45/0.9067	31.36/0.8879	29.50/0.8946
	FSRCNN [9]	12K	6.0G	37.05/0.9560	32.66/0.9090	31.53/0.8920	29.88/0.9020
	VDSR [18]	665K	612.6G	37.53/0.9590	33.05/0.9130	31.90/0.8960	30.77/0.9140
	DRCN [19]	1,774K	17,974.3G	37.63/0.9588	33.04/0.9118	31.85/0.8942	30.75/0.9133
	LapSRN [20]	813K	29.9G	37.52/0.9591	33.08/0.9130	31.80/0.8950	30.41/0.9101
	DRRN [29]	297K	6,796.9G	37.74/0.9591	33.23/0.9136	32.05/0.8973	31.23/0.9188
	MemNet [30]	677K	2,662.4G	37.78/0.9597	33.28/0.9142	32.08/0.8978	31.31/0.9195
	IDN* [17]	579K	124.6G	37.85/0.9598	33.58/0.9178	32.11/0.8989	31.95/0.9266
	CARN [2]	1,592K	222.8G	37.76/0.9590	33.52/0.9166	32.09/0.8978	31.92/0.9256
	BSRN [5]	594K	1666.7G	37.78/0.9591	33.43/0.9155	32.11/0.8983	31.92/0.9261
	FALSR-A [6]	1,021K	234.7G	37.82/0.9595	33.55/0.9168	32.12/0.8987	31.93/0.9256
	FALSR-B [6]	326k	74.7G	37.61/0.9585	33.29/0.9143	31.97/0.8967	31.28/0.9191
	FALSR-C [6]	408k	93.7G	37.66/0.9586	33.26/0.9140	31.96/0.8965	31.24/0.9187
	IMDN [16]	694K	158.8G	38.00/0.9605	33.63/0.9177	32.19/0.8996	32.17/0.9283
	LatticeNet (Ours)	756K	169.5G	38.15/0.9610	33.78/0.9193	32.25/0.9005	32.43/0.9302
3×	Bicubic	-	-	30.39/0.8682	27.55/0.7742	27.21/0.7385	24.46/0.7349
	SRCNN [8]	57K	52.7G	32.75/0.9090	29.30/0.8215	28.41/0.7863	26.24/0.7989
	FSRCNN [9]	12K	5.0G	33.18/0.9140	29.37/0.8240	28.53/0.7910	26.43/0.8080
	VDSR [18]	665K	612.6G	33.67/0.9210	29.78/0.8320	28.83/0.7990	27.14/0.8290
	DRCN [19]	1,774K	17,974.3G	33.82/0.9226	29.76/0.8311	28.80/0.7963	27.15/0.8276
	DRRN [29]	297K	6,796,9G	34.03/0.9244	29.96/0.8349	28.95/0.8004	27.53/0.8378
	MemNet [30]	677K	2,662.4G	34.09/0.9248	30.00/0.8350	28.96/0.8001	27.56/0.8376
	IDN* [17]	588K	56.3G	34.24/0.9260	30.27/0.8408	29.03/0.8038	27.99/0.8489
	CARN [2]	1,592K	118.8G	34.29/0.9255	30.29/0.8407	29.06/0.8034	28.06/0.8493
	BSRN [5]	779K	761.1G	34.32/0.9255	30.25/0.8404	29.07/0.8039	28.04/0.8497
	IMDN [16]	703K	71.5G	34.36/0.9270	30.32/0.8417	29.09/0.8046	28.17/0.8519
	LatticeNet (Ours)	765K	76.3G	34.53/0.9281	30.39/0.8424	29.15/0.8059	28.33/0.8538
4×	Bicubic	-	-	28.42/0.8104	26.00/0.7027	25.96/0.6675	23.14/0.6577
	SRCNN [8]	57K	52.7G	30.48/0.8628	27.50/0.7513	26.90/0.7101	24.52/0.7221
	FSRCNN [9]	12K	4.6G	30.72/0.8660	27.61/0.7550	26.98/0.7150	24.62/0.7280
	VDSR [18]	665K	612.6G	31.35/0.8830	28.02/0.7680	27.29/0.7260	25.18/0.7540
	DRCN [19]	1,774K	17,974.3G	31.53/0.8854	28.02/0.7670	27.23/0.7233	25.14/0.7510
	LapSRN [20]	813K	149.4G	31.54/0.8850	28.19/0.7720	27.32/0.7270	25.21/0.7560
	DRRN [29]	297K	6,796.9G	31.68/0.8888	28.21/0.7720	27.38/0.7284	25.44/0.7638
	MemNet [30]	677K	2,662.4G	31.74/0.8893	28.26/0.7723	27.40/0.7281	25.50/0.7630
	IDN* [17]	600K	32.3G	31.99/0.8928	28.52/0.7794	27.52/0.7339	25.92/0.7801
	CARN [2]	1,592K	90.9G	32.13/0.8937	28.60/0.7806	27.58/ 0.7349	26.07/0.7837
	BSRN [5]	742K	451.8G	32.14/0.8937	28.56/0.7803	27.57/0.7353	26.03/0.7835
	IMDN [16]	715K	40.9G	32.21/0.8948	28.58/0.7811	27.56/0.7353	26.04/0.7838
	LatticeNet (Ours)	777K	43.6G	32.30/0.8962	28.68/0.7830	27.62/0.7367	26.25/0.7873

4.5 Comparisons with the State-of-the-arts

In this section, we conduct extensive experiments on four publicly available SR benchmark datasets and compare with 12 state-of-the-art lightweight SR models:
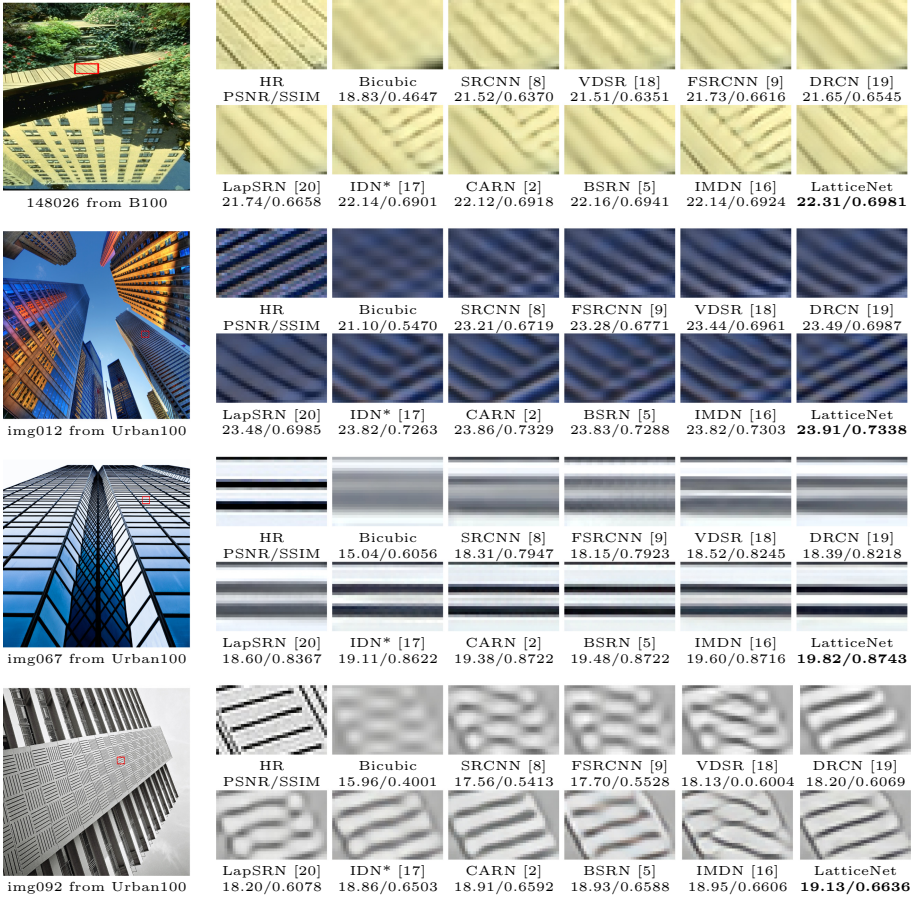
Fig. 7. Visual comparisons of the state-of-the-art lightweight methods and our LatticeNet on B100 and Urban100 for 4× SR. **Zoom in for best view**

SRCNN [8], FSRCNN [9], VDSR [18], DRCN [19], LapSRN [20], DRRN [29], MemNet [30], IDN* [17] [1], CARN [2], BSRN [5], FALSR [6], and IMDN [16].

The quantitative comparisons for 2×, 3×, and 4× SR are shown in Table 6. When compared with all given methods, our LatticeNet performs the best on the four datasets. Meanwhile, we also give the network parameters for all the comparison methods. Our model is at the medium level of the model complexity with less 800K parameters and the performance is comparable with CARN and IMDN. Besides, Mult-Adds of LatticeNet is also relatively lower. It demonstrates that our method is superior to other SR methods in comprehensive performance.

---

[1] IDN* refers to the results given in https://github.com/Zheng222/IDN-tensorflow, which is retrained on DIV2K dataset.

The visual comparisons on scale 4× on B100 and Urban100 are depicted in Fig. 7. For Image "148026" in B100, our method can recover the correct texture of the wooden bridge well than other methods. Besides, for image "img044" and "img067" in Urban100 dataset, we can also observe that our results are more favorable and can recover more details. Though our method achieves comparable performance against IMDN in PSNR and SSIM, our method is obviously superior to IMDN with a large margin in the visual effect.

5   Conclusions

In this paper, we propose the lattice block which is an economical structure favoring the lightweight model design. It has the potential of multiple combinations of two RBs benefited by the combination coefficients. The combination coefficients of the lattice block are learned with the attention mechanism for better SR performance. It can embed in any SR networks using RBs as basic block. Based on the lattice block, we also build LatticeNet which is a lightweight SR model. Moreover, we adopt a backward sequential concatenation strategy to integrate more contextual information from different receptive fields. Experimental results on several benchmark datasets also demonstrate that our method can achieve superior performance with moderate parameters.

References

1. Agustsson, E., Timofte, R.: NTIRE 2017 challenge on single image super-resolution: dataset and study. In: CVPRW (2017)
2. Ahn, N., Kang, B., Sohn, K.A.: Fast, accurate, and lightweight super-resolution with cascading residual network. In: ECCV (2018)
3. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. TPAMI **33**, 898–916 (2011)
4. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-Complexity Single-Image Super-resolution Based on Nonnegative Neighbor Embedding. BMVA Press (2012)
5. Choi, J.H., Kim, J.H., Cheon, M., Lee, J.S.: Lightweight and efficient image super-resolution with block state-based recursive network. arXiv:1811.12546 (2018)
6. Chu, X., Zhang, B., Ma, H., Xu, R., Li, J., Li, Q.: Fast, accurate and lightweight super-resolution with neural architecture search. arXiv:1901.07261 (2019)
7. Dai, T., Cai, J., Zhang, Y., Xia, S.T., Zhang, L.: Second-order attention network for single image super-resolution. In: CVPR (2019)
8. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. TPAMI **38**, 295–307 (2016)
9. Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 391–407. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_25

10. Fu, J., Liu, J., Tian, H., Fang, Z., Lu, H.: Dual attention network for scene segmentation. arXiv:1809.02983 (2018)
11. Gao, Q., Zhao, Y., Li, G., Tong, T.: Image super-resolution using knowledge distillation. In: Jawahar, C.V., Li, H., Mori, G., Schindler, K. (eds.) ACCV 2018. LNCS, vol. 11362, pp. 527–541. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20890-5_34
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
13. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. arXiv:1603.05027 (2016)
14. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018)
15. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: CVPR (2015)
16. Hui, Z., Gao, X., Yang, Y., Wang, X.: Lightweight image super-resolution with information multi-distillation network. In: ACMMM (2019)
17. Hui, Z., Wang, X., Gao, X.: Fast and accurate single image super-resolution via information distillation network. In: CVPR (2018)
18. Kim, J., Kwon Lee, J., Mu Lee, K.: Accurate image super-resolution using very deep convolutional networks. In: CVPR (2016)
19. Kim, J., Kwon Lee, J., Mu Lee, K.: Deeply-recursive convolutional network for image super-resolution. In: CVPR (2016)
20. Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Deep Laplacian pyramid networks for fast and accurate super-resolution. In: CVPR (2017)
21. Ledig, C., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR (2017)
22. Li, B., Gao, X.: Lattice structure for regular linear phase paraunitary filter bank with odd decimation factor. IEEE Sig. Process. Lett. **21**, 14–17 (2014)
23. Li, X., Wu, J., Lin, Z., Liu, H., Zha, H.: Recurrent squeeze-and-excitation context aggregation net for single image deraining. In: ECCV (2018)
24. Li, Z., Yang, J., Liu, Z., Yang, X., Wu, W.: Feedback network for image super-resolution. In: CVPR (2019)
25. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. In: CVPRW (2017)
26. Lin, T.Y., Dollr, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection (2017)
27. Shi, W., et al.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: CVPR (2016)
28. Shrivastava, A., Sukthankar, R., Malik, J., Gupta, A.: Beyond skip connections: top-down modulation for object detection. arXiv:1612.06851 (2016)
29. Tai, Y., Yang, J., Liu, X.: Image super-resolution via deep recursive residual network. In: CVPR (2017)
30. Tai, Y., Yang, J., Liu, X., Xu, C.: MemNet: a persistent memory network for image restoration. In: ICCV (2017)
31. Wang, W., Guo, R., Tian, Y., Yang, W.: CFSNet: toward a controllable feature space for image restoration. In: ICCV (2019)
32. Woo, S., Park, J., Lee, J., Kweon, I.: CBAM: convolutional block attention module. arXiv:1807.06521
33. Qiu, Y., Wang, R., Tao, D., Cheng, J.: Embedded block residual network: a recursive restoration model for single-image super-resolution. In: ICCV (2019)

34. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: International Conference on Curves and Surfaces (2010)
35. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: ECCV (2018)
36. Zhang, Y., Tian, Y., Kong, Y., Zhong, B., Fu, Y.: Residual dense network for image super-resolution. In: CVPR (2018)