Information Sciences 516 (2020) 220–233

Contents lists available at ScienceDirect

Information Sciences

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

Lightweight group convolutional network for single image super-resolution

Aiping Yang[a], Bingwang Yang[a], Zhong Ji[a,*], Yanwei Pang[a], Ling Shao[a,b]

[a] School of Electrical and Information Engineering, Tianjin University, Tianjin, China
[b] Inception Institute of Artificial Intelligence, Abu Dhabi, UAE

A R T I C L E I N F O

Article history:
Received 14 January 2019
Revised 30 November 2019
Accepted 25 December 2019
Available online 26 December 2019

Keywords:
Image super-resolution
Convolutional neural network
Group convolution
Channel attention
Lightweight network

A B S T R A C T

Recently, deep learning methods have demonstrated significant reconstruction performance on Single Image Super-Resolution (SISR). However, most of them demand a huge amount of computational and memory consumption and hard to be applied to real-world applications. To this end, we propose a fast Lightweight Group Convolution Network (LGCN) model for SISR to alleviate this problem. Specifically, we develop a cascaded memory group convolutional network for SISR, which cascades several Memory Group Convolutional Networks (MGCN). There are two main merits on MGCN. One is that it consists of several group convolutional layers and 1×1 convolutional layers with densely connected structure. The group convolution is utilized to reduce the parameters of LGCN, and the 1×1 convolution is not only used to create a linear combination of the output of group convolutional layer, but also to gather local information progressively. The other one is that it utilizes channel attention unit to model channel-wise relationships to improve performance. Experimental results on four popular datasets show that the proposed LGCN not only outperforms the state-of-the-art SISR methods, but also achieves faster speed.

1. Introduction

Single Image Super-Resolution (SISR) has received increasing research attention for decades due to its great value in domains of image surveillance, object recognition and medical imaging [1–5]. It aims at reconstructing a high-resolution (HR) image with high-frequency information from a low-resolution (LR) image. However, SISR is a typical ill-posed inverse problem and has no unique solution, which makes it a challenging task.

With the renaissance of the neural network, deep learning methods, especially Convolutional Neural Networks (CNN), dramatically push forward the development of SISR [6,7]. It should be noticed that SISR is quite different from image classification task [8], thus the off-the-shelf CNN models cannot be directly applied to SISR. For example, it is obvious that the pooling layer is harmful to SISR. Thus, special design is required for CNN-based SISR methods. In recent years, we have witnessed their successes, which significantly boost the performance of SISR. For example, as one of the pioneering work in this line, SRCNN [6] employs a three-layer CNN to learn the mapping from LR to HR pairs by end-to-end optimizing the feature extraction, non-linear mapping and image reconstruction stages. Kim et al. [7] proposed a VDSR model containing

* Corresponding author.
E-mail address: jizhong@tju.edu.cn (Z. Ji).

https://doi.org/10.1016/j.ins.2019.12.057
0020-0255/© 2019 Elsevier Inc. All rights reserved.

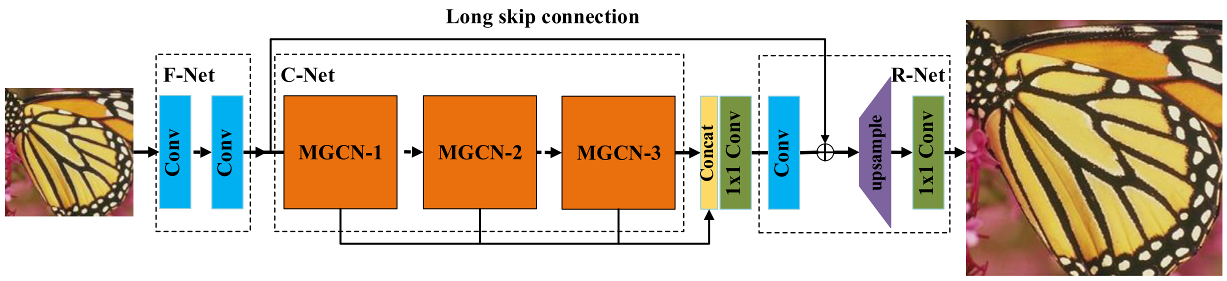


Fig. 1. Architecture of the proposed LGCN model. It consists three main components: Feature Extraction Net (F-Net), Cascaded Memory Group Convolutional Net (C-Net) and Reconstruction Net (R-Net). MGCN stands for Memory Group Convolutional Net.

20 CNN layers, which applies residual learning and adjustable gradient clipping to accelerate convergence. Lim et al. [9] presented an EDSR model based on residual block without batch normalization layers, which builds a very wide network by using simplified residual blocks to promote the performance.

However, although this line of methods achieves good performance, with the deepening and widening of network layers, the amount of parameters increases greatly. This results in an increasing computational and memory burden, which makes the methods less applicable in practice. To this end, lightweight and accurate SISR method is becoming one of the promising directions to reduce the model parameters while keeping the performance. For example, Kim et al. [10] constructed a Deeply-Recursive Convolutional Network (DRCN) by adopting recursive neural network to reduce the model parameters. Further, Tai et al. [11] added a residual architecture to DRCN to train a 52-layer recursive neural network for better performance, which is named Deep Recursive Residual Network (DRRN). However, both methods still have some drawbacks. First, they upsample the input images to the desired resolution before feeding it to network for prediction, which increases the computational cost and requires a large amount of memory. Second, DRRN increases the depth of the network to compensate the accuracy loss due to using a recursive network, which increases the inference time.

To this end, we propose a Lightweight Group Convolution Network (LGCN) for SISR, as illustrated in Fig. 1. It consists three main components: Feature Extraction Net (F-Net), Cascaded Memory Group Convolutional Net (C-Net) and Reconstruction Net (R-Net). Our contributions focus on the C-Net, which cascades multiple Memory Group Convolutional Nets (MGCN). Specifically, MGCN contains a memory unit and a channel attention unit. The memory unit consists of several alternating group convolutional layers and 1×1 convolutional layers with densely connected structure to build the memory mechanism. Different from DenseNet [12], we utilize 1×1 convolution as the middle layer to gather local information progressively. Thus the 1×1 convolutional layer in MGCN can not only create a linear combination of the output of group convolutional layer, but also gather local information progressively. The channel attention unit is inspired by the idea of squeeze and excitation net [13], however, we replace the fully connection layer in squeeze and excitation net [13] with 1×1 convolutional layer to better model channel-wise relationships.

The main contributions are summarized as follows. First, we propose LGCN, a CNN-based SISR method with cascaded modules, as illustrated in Fig. 1. It achieves both high performance and less computation on benchmark SISR datasets. Second, we develop a Memory Group Convolutional Network (MGCN), which uses group convolution as the convolutional layer to reduce the network parameters. It should be noted that few works have been proposed to investigate the effect of group convolution for low-level vision tasks (e.g., SISR). In addition, we develop an original unit named Memory unit that employs 1×1 convolution with densely connected structure to aggregate multi-level features efficiently for memory mechanism. The 1×1 convolutional layers in Memory unit have two merits: (1) They can create a linear combination of the output of group convolutional layer. (2) Different from SRDenseNet [14] that takes the result of concatenation operations as the input of 3×3 convolutional layer directly, we take 1×1 convolutional layer as the middle layer in MGCN to gather local information progressively. In this way, it can reduce the increase of parameters. Third, a novel channel attention method for SISR in MGCN is presented to better recalibrate channel-wise features, which replaces the fully connection layer with 1×1 convolutional layer based on the idea of squeeze and excitation [13]. Further, we cascade multiple MGCNs to collect more contextual information. Extensive experiments on benchmark datasets show our LGCN model achieves competitive results against state-of-the-art methods on both accuracy and speed, as shown in Fig. 2.

The rest of the paper is organized as follows: Section 2 discusses the related work, includes SISR methods and compact neural models. Section 3 describes the proposed LGCN for SISR in detail. Model analysis and experimental comparison against several state-of-the-art methods are presented in Section 4. Section 5 concludes the paper.

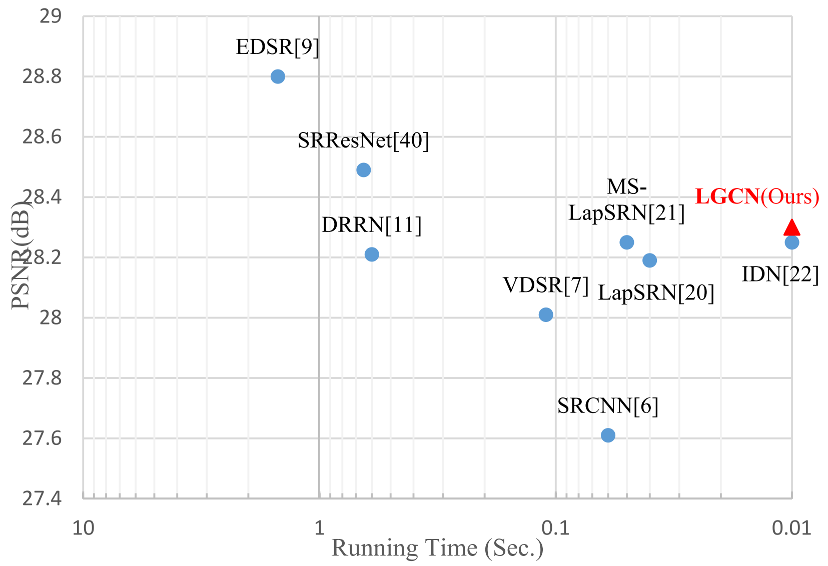


Fig. 2. Running time and performance trade-off. The results are evaluated on Set14 with the scaling factor 4 ×. The proposed LGCN is faster than other comparative methods and achieves competing performance.

2. Related work

2.1. Lightweight CNN-based SISR methods

In recent years, CNN-based methods have dominated the research of SISR. Since the first method, SRCNN [6], was proposed, several new methods have been developed. For example, Mao et al. [15] proposed an encoder-decoder network with symmetric skip connections to restore the high-resolution image. Inspired by ResNet [16], Lim et al. [9] proposed EDSR model by establishing a very deep network and removing the batch normalization layers from the residual block. Haris et al. [17] proposed DBPN model, which exploits iterative upsampling and downsampling layers to provide an error feedback mechanism for projection errors at each stages.

However, one of the largest bottlenecks for these methods to be used in practice is the huge computational burden. This is because that deep networks generally demand large amount of parameters. To make the CNN-based SISR methods more practicable, researchers have paid their attention to the design of compact and lightweight networks. As one of the earlier attempts, FSRCNN [18] extracts feature maps in LR space and upsamples images by deconvolution operation in the last layer to accelerate SRCNN [6], while ESPCN [19] replaces the deconvolution operation with an efficient sub-pixel convolution for real-time processing. Afterwards, Kim et al. [10] proposed a deeply-recursive convolutional network (DRCN), which utilizes a recursive network with skip connection to increase depth of model without introducing additional parameters. Tai et al. [11] proposed Deep Recursive Residual Network(DRRN) by combining recursive and residual network to improve DRCN. However, these methods still require heavy computation resources because of the very deep network.

To this end, Lai et al. [20] proposed a fast SISR method known as LapSRN, which introduces the Laplacian pyramid architecture to progressively predict the sub-band residuals in a coarse-to-fine fashion. The authors also extended the LapSRN to MS-LapSRN [21] by introducing the recursive layers to share parameters across as well as within pyramid level to reduce the number of parameters. Recently, Hui et al. [22] proposed a fast and accurate SISR method named Information Distillation Network(IDN) by introducing an information distillation block to extract the local long and short-path features efficiently. Ahn et al. [23] proposed a Cascading Residual Network (CARN), which combines an efficient residual block with cascading mechanism and recursive network scheme to achieve a lightweight model for SISR. Our method is similar to the CARN [23], the difference between our model and CARN is that we do not employ the residual block in our MGCN unit and we build a more concise structure by connecting the output of 1 × 1 convolutional layer while the CARN cascades the output of the residual block. Moreover, we utilize the hierarchical feature fusion and long skip connection in a global way while the CARN utilizes global cascading mechanism.

2.2. Compact models

Recently, several small and efficient neural networks have been proposed for classification and detection tasks [24–26]. For example, SqueezeNet [24] built an efficient neural network based on AlexNet [8] by replacing 3 × 3 convolution with 1 × 1 convolution, and achieves the AlexNet-level accuracy with 50 × fewer parameters. ResNeXt [25] replaces the regular 3 × 3 convolutional layers in residual blocks of ResNet [16] with group convolutional layers to demonstrate the effectiveness of the group convolution that was first introduced in AlexNet [8]. MobileNet [26] applies depth-wise separable convolution to factorize a standard convolution into a depthwise convolution and a 1 × 1 convolution to build a lightweight deep neural network and show the effectiveness of such architecture across a wide range of application. ShuffleNet [27] utilizes group convolutions and pointwise convolutions and introduces channel shuffle operations to greatly reduce computational cost while maintaining the accuracy. Recently, Hu et al. proposed SENet [13] to explicitly model interdependence among channels and reweight channel-wise feature to improve the performance of networks with slight increase in computational cost. Our proposed network is inspired by the ResNext [25] and SENet [13], where we apply the ideas of group convolution and channel attention in SENet [13] in our MGCN unit. However, different from them, the proposed MGCN unit contains several alternating group convolutional layers and 1 × 1 convolutional layers with densely connected structure to efficiently extract the features. And we also replace the fully connection layer in SENet with 1 × 1 convolutional layer to build a new channel attention unit to better recalibrate channel-wise features. In this way, the proposed LGCN method is ensured to be both fast and accurate for SISR.

3. Proposed LGCN model

3.1. Network structure

As shown in Fig. 1, our proposed LGCN mainly consists three sub-nets: F-Net, C-Net and R-Net. We utilize the hierarchical feature fusion and long skip connection in a holistic way to combine low-level and high-level features to learn abundant information for super-resolution reconstruction. Different from MemNet [28] and other existing methods, our model takes the low-resolution images as input directly and employs a deconvolutional layer to upsample images. Here, we denote I_{LR} and I_{SR} as the input and output of LGCN, respectively. As for the F-Net, we design a neural network with two convolutional layers to extract the visual features from the original LR image. The procedure can be expressed as:

$$F_0 = H_F(I_{LR}), \tag{1}$$

where $H_F(\cdot)$ represents the nonlinear function in F-Net and F_0 denotes the extracted visual feature. It is then used as the input of the next sub-net, C-Net.

The C-Net cascades several Memory Group Convolutional Networks. Suppose we have **K** MGCNs, the process of C-Net can be formulated as:

$$F_k = H_{MGCN,k}(H_{MGCN,k-1}(\cdots(H_{MGCN,1}(F_0))), \qquad k = 1, \ldots, \mathbf{K}. \tag{2}$$

where F_k denotes the output of the kth MGCN, and $H_{MGCN,k}(\cdot)$ denotes the operation of k-th proposed MGCN, which contains both memory unit and channel attention unit. The proposed MGCN can efficiently utilize the features from all convolutional layers within the MGCN. More detail will be introduced in Section 3.2.

Then, we utilize hierarchical feature fusion and long skip connection to make full use of the hierarchical features produced by MGCNs, which can be formulated as:

$$F_G = H_{HF}([F_1, \ldots\ldots\ldots F_k \ldots\ldots\ldots F_K]) + F_0, \tag{3}$$

where F_G is the output feature maps of the operations of both hierarchical feature fusion and long skip connection. $[F_1, \ldots sF_k \ldots sF_K]$ denotes the concatenation of feature maps produced by MGCN from 1 to **K**. The analysis of the number of MGCN will be shown in Section 4. $H_{HF}(\ldots)$ denotes the operation of the hierarchical feature fusion, which consists of a 1 × 1 convolutional layer and a 3 × 3 convolutional layer. The 1 × 1 convolutional layer is used to combine the features from previous MGCN. The following 3 × 3 convolutional layer is then applied to further extract the features. Hierarchical feature fusion can further adaptively fuse the features produced by MGCNs to effectively carry high-level frequency signals from shallow to deep layers to learn a variety of information from different receptive fields. Besides, by incorporating long skip connection, our model can learn the residual between the deep feature maps and shallow feature maps, the residual is sparsely and easily learned, leading to better convergence.

Finally, in R-Net, we take a transposed convolutional layer followed by a convolutional layer as the reconstruction block. To this end, we utilize a transposed convolutional layer to generate high resolution feature. It can be expressed as:

$$F_{UP} = H_{UP}(F_G), \tag{4}$$

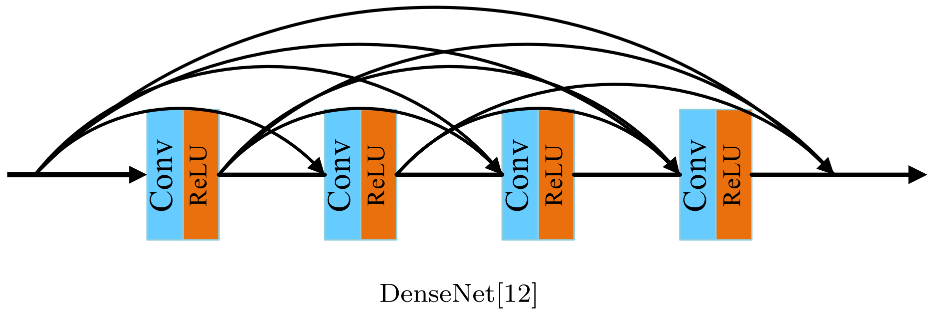
where F_{UP} and $H_{UP}(\ldots)$ denote upscaled feature and a transposed convolutional layer, respectively. The final output of the LGCN model is then reconstructed by one convolutional layer:

$$I_{SR} = H_{REC}(F_{UP}) = H_{LGCN}(I_{LR}), \tag{5}$$

where $H_{REC}(\ldots)$ and $H_{LGCN}(\ldots)$ denote the reconstruction layer and the function of our LGCN, respectively.

3.2. C-Net

In this section, we will introduce the detail of C-Net, which cascades several MGCN. The structure of MGCN is shown in Fig. 3(b). It contains a memory unit and channel attention unit. The LR features firstly pass through memory unit to further extract features efficiently, then the output feature maps of the memory unit are reweighted by channel attention unit to model the interdependencies among channels to further improve the model's representational ability.



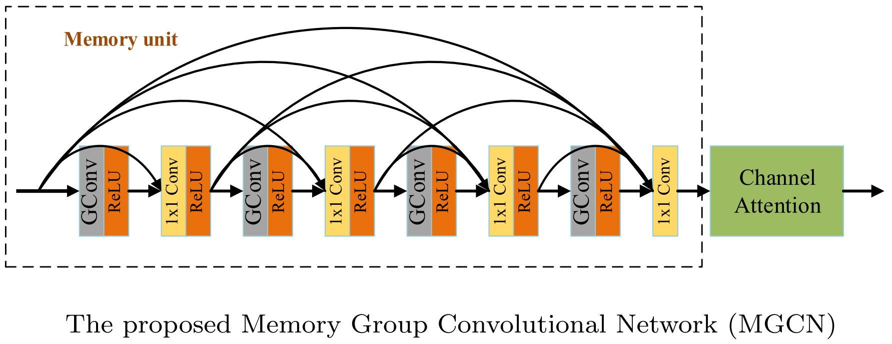


Fig. 3. Architecture comparison for the proposed DenseNet (a) and MGCN (b). Different from DenseNet (a), MGCN utilizes 1 × 1 convolutional layer as the middle layer to gather the preceding information, and the channel attention unit is applied to better model the interdependencies among channel.

3.2.1. Memory unit

As shown in Fig. 3(b), our memory unit consists of several alternant group convolutional layers and 1 × 1 convolutional layers with densely connected structure to transmit the state of the all preceding layers to current layer to build the memory mechanism.

(a) Memory mechanism. The outputs of the intermediary convolutional layer are cascaded into the high layers to formulate a memory mechanism. As shown in Fig. 3(b), different from DenseNet (Fig. 3(a)), we utilize 1 × 1 convolutional layer to gather the information from all the preceding layers. Let F_d be the output of memory unit, F_{k-1} be the input of the MGCN, the output of the jth group convolutional layer in memory unit of the kth MGCN can be formulated as:

$$\begin{aligned} F_g^{k,1} &= \tau(W_g^{k,1}[F_{k-1}]), \\ F_g^{k,j} &= \tau(W_g^{k,j}[F_1^{k,j-1}]), \qquad j = 2, \ldots, \mathbf{J}. \end{aligned} \tag{6}$$

where τ denotes the activation function, and $\mathbf{J}$ denotes the number of the group convolutional layers, $W_g^{k,j}$ denotes the weights of the jth group convolutional layer, and $F_1^{k,j-1}$ denotes the output of (j-1)-th 1x1 convolutional layer. Further, the output of ith 1 × 1 convolutional layer in memory unit of kth MGCN can be defined as:

$$\begin{aligned} F_1^{k,1} &= \tau(W_1^{k,1}[F_{k-1}, F_g^{k,1}]), \\ F_1^{k,i} &= \tau(W_1^{k,i}[F_{k-1}, F_1^{k,1}, F_1^{k,2} \ldots\ldots\ldots, F_1^{k,i-1}, F_g^{k,i}]), \qquad i = 2, \ldots, \mathbf{I} - 1. \end{aligned} \tag{7}$$

where $W_1^{k,i}$ is the weights of the ith 1 × 1 convolutional layer, τ denotes the activation function, and [ ] denotes the operation of concatenation, $\mathbf{I}$ denotes the number of the 1 × 1 convolutional layers. The output F_d can be obtained:

$$F_d = F_1^{k,I} = W_1^{k,I}[F_{k-1}, F_1^{k,1}, F_1^{k,2} \ldots\ldots\ldots, F_1^{k,I-1}, F_g^{k,I}]. \tag{8}$$

The 1 × 1 convolution with densely connected structure enables our memory unit incorporating features from preceding layers, which allows to learn the multi-level representations. And this connection strengthens the flow of information for quick information propagation through network. In addition, this connection can substantially reduce the number of parameters by reusing previous features, which makes our model more efficient. Specifically, the 1 × 1 convolutional layer not only aggregates multi-level features for our unit, but creates a linear combination of the output of group convolutional layer. *(b) Group convolution*. As shown in Fig. 4(b), group convolution significantly reduces the parameters comparing with standard convolution. Based on this structure, we utilize the group convolution in our model. Assuming D_k is the kernel size of the group convolutional layer, and m and n are the numbers of the input and output channels of the group convolutional

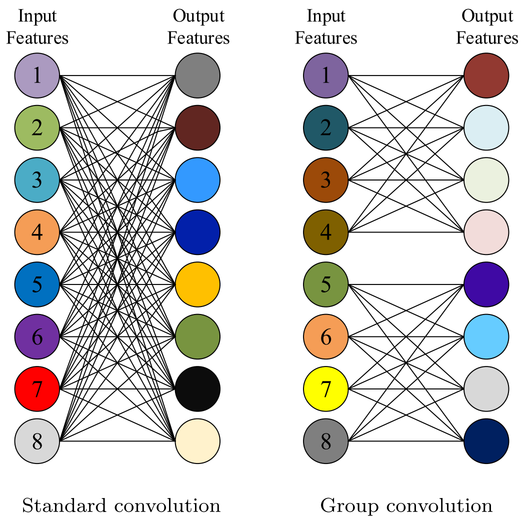


Fig. 4. Structure comparison for standard convolution (a) and group convolution (group size = 2) (b). Each filter in (a) is convolved with all the previous layer's feature maps, while each filter in (b) is convolved with only half the previous layer's feature maps.

layer, respectively. Because of the usage of padding, the input and output feature size of the group convolutional layer are the same, we assume D_F is the input and output feature size.

We provide the comparison of computational cost by using standard convolution and the group one in the following. Note that we only count the cost of convolutional layers and ignore the cost of the addition or activation because both memory unit with standard convolution and memory unit with group convolution have the same amount of cost in terms of addition and activation. When utilizing standard convolution, the computational cost of the memory unit is as $4 \times (D_k \times D_k \times m \times n \times D_F \times D_F) + (2 \times n + 3 \times n + 4 \times n + 5 \times n) \times m \times D_F \times D_F$. It is the sum of the parameters of four $D_k \times D_k$ standard convolutions and four 1×1 convolutions. When applying to group convolution, the computational cost of the memory unit is as $4 \times (D_k \times D_k \times m \times \frac{n}{G} \times D_F \times D_F) + (2 \times n + 3 \times n + 4 \times n + 5 \times n) \times m \times D_F \times D_F$, where G denote the group size. Thus, the reduction ratio of computation is:

$$\frac{(4(D_k \times D_k \times m \times \frac{n}{G} + (2 \times n + 3 \times n + 4 \times n + 5 \times n) \times m) \times D_F \times D_F}{(4(D_k \times D_k \times m \times n + (2 \times n + 3 \times n + 4 \times n + 5 \times n) \times m) \times D_F \times D_F} = \frac{4D_k^2 + 14G}{4GD_k^2 + 14G}. \tag{9}$$

Our model uses 3×3 group convolutions, that is D_k=3. In addition, G can be set from 2 to 64, which means the corresponding computational cost may reduce between 1.6 to 3.4 times depending on the group size. The analysis of the group size will be shown in Section 4.

3.2.2. Channel attention unit

Previous CNN based SISR methods treat LR channel-wise features equally, which is not flexible for real case. Inspired by SENet [13], we build a channel attention unit based on the conception of SE module, as shown in Fig. 5. Channel attention unit enables our model to focus on more informative features by emphasizing important features and suppressing useless features among channels. We firstly take the global spatial information into a channel descriptor. Taking $U = [u_1, \ldots su_c, \ldots s, u_C]$ as an input, which contains C feature maps with size $H \times W$. We utilize global average pooling to

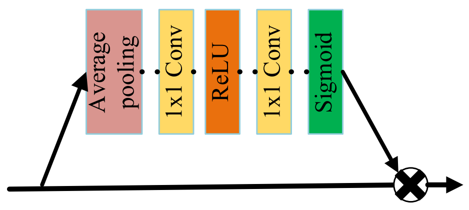


Fig. 5. Channel Attention unit.

generate channel-wise statistics z, where the cth element of z is calculated by:

$$z_c = g(u_c) = \frac{1}{H \times W} \sum_{i}^{H} \sum_{j}^{W} u_c(i, j), \tag{10}$$

where $g(\ldots)$ denotes global pooling function, $u_c(i, j)$ denotes the value at position (i, j) of cth feature u_c.

In order to fully capture channel-wise dependencies from the information aggregated in global average pooling operation, we employ a gating mechanism with a sigmoid activation:

$$\alpha = \sigma(W_u \delta(W_d z)), \tag{11}$$

where σ and δ denote Sigmoid function and ReLU [29] function, respectively. In contrast to original SENet [13], we utilize 1 × 1 convolutional layer instead of fully connected layer as illustrated in Fig. 5. W_d denotes the weights of a convolutional layer, which reduces the dimensionality with reduction ratio r. After being activated by ReLU, the low-dimension signal is then increased with ratio r by a dimensionality-increasing layer with weights W_u. The α denotes the final channel statistics, and can be used to rescale the corresponding channels of the input:

$$\widehat{u}_c = F_{scale}(u_{c,}\alpha_c) = \alpha_c \ldots u_c, \tag{12}$$

where the output of channel attention unit $\widehat{U} = [\widehat{u}_1, \ldots\ldots\ldots \widehat{u}_c, \ldots\ldots\ldots, \widehat{u}_C]$, $F_{scale}(u_{c,}\alpha_c)$ denotes channel-wise multiplication between the feature map u_c and scaling factor α_c in cth channel. Thus, the output of the proposed MGCN F_k can be formulated with:

$$F_k = F_{scale}(F_d, \alpha), \tag{13}$$

where F_d denotes the output for memory unit. With Channel Attention unit, noise information in previous feature maps can be reduced, and the informative features will receive more attention. Thus, the feature discriminative ability is boosted. Its effectiveness will be shown in the section of ablation studies.

3.3. Loss function

Following the previous works [22,30], we utilize L_1 loss function to optimize our LGCN. Given **N** predicted high-resolution image I_{SR} and the corresponding ground-truth high-resolution image I_{HR}, they can be formulated as follows:

$$L_1 = \frac{1}{N} \sum_{i}^{N} \left\| I^i_{HR} - I^i_{SR} \right\|_1. \tag{14}$$

4. Experiments

In this section, we first introduce the experimental settings and the implementation details of our proposed SISR method. We then compare our method against several state-of-the-art models on four benchmark datasets in terms of reconstruction performance and speed. Finally, we provide the ablation study for different components.

4.1. Experimental settings

Following the popular settings [20,22], we train our LGCN and other comparative models with 91 images from Yang et al. [31] and 200 images from Berkeley Segmentation Dataset(BSD) [32], respectively. We augment the data in three ways: (1) Rotate the images with degree of 90°, 180°, 270°. (2) Flip images horizontally. (3) Downscale the images with factor of 0.9, 0.8, 0.7 and 0.6. For testing, we evaluate them on four widely used benchmark datasets: Set5 [33], Set14 [34], BSD100 [35] and Ubran100 [36], respectively. All the RGB images of the four benchmark datasets are converted into the Ycbcr color space. Following [20,22], we only process the Y-channel, and simply enlarge the rest color components with Bicubic interpolation. To produce LR images, we downscale the high-resolution(HR) images on particular scaling factors using Bicubic interpolation.

In each training batch, we randomly sample 64 patches. For different upscaling factor 2 ×, 3 ×, 4 ×, we firstly train our model with $29^2/58^2$, $15^2/45^2$, $11^2/44^2$ LR/HR image patches, then we use $35^2/70^2$, $25^2/75^2$, $19^2/76^2$ LR/HR patches for first fine-tuning. At last, we use $64^2/128^2$, $43^2/129^2$, $32^2/128^2$ LR/HR image patches to fine-tune again. Our model is trained by ADAM optimizer [37] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate is initialized to $1e-4$ and decreases by a factor 10 for every 2.5×10^5 iterations in total 5×10^5 iterations in training phases. On the other hand, the learning rate is set to $1e-5$ for total 2×10^5 iterations for two fine-tuning phases. We initialize the weights by using the method in [38] and the bias is set to zero. We build our model using PyTorch on a NVIDIA TITAN XP GPU.

We set the number of group convolutional layers and 1 × 1 convolutional layers in MGCN as $\mathbf{I} = \mathbf{J} = 4$, and 3 × 3 as the size of all convolutional layers in F-Net. We utilize zero-padding to keep size fixed for all 3 × 3 convolutional layers and the transposed convolutional layers. All convolutional layers are followed by ReLU [29] except for the last 1 × 1

Table 1
Quantitative results against the state-of-art SISR algorithms: average PSNR/SSIM for scaling factors 2 $\times$, 3 $\times$ and 4 $\times$, respectively. The best performances are marked in bold, and the second best performances are marked with underline.

Algorithm	Scale	Set5 PSNR/SSIM	Set14 PSNR/SSIM	BSD 100 PSNR/SSIM	Urban 100 PSNR/SSIM
Bicubic	2	33.66/0.930	30.24/0.869	29.56/0.843	26.88/0.841
SRCNN [6]	2	36.66/0.954	32.45/0.906	31.36/0.888	29.50/0.895
VDSR [7]	2	37.53/0.958	33.05/0.913	31.90/0.896	30.77/0.914
LapSRN [20]	2	37.52/$\underline{0.959}$	33.08/0.913	31.80/0.895	30.41/0.910
DRRN [11]	2	37.74/$\underline{0.959}$	33.23/0.913	$\underline{32.05}$/0.897	$\underline{31.23}$/0.919
MS-LapSRN [21]	2	37.72/$\underline{0.959}$	33.24/$\underline{0.914}$	32.00/0.898	31.01/0.917
IDN [22]	2	**37.83**/**0.960**	$\underline{33.30}$/$\underline{0.914}$	**32.08**/$\underline{0.899}$	**31.27**/$\underline{0.920}$
LGCN (Ours)	2	$\underline{37.75}$/**0.960**	**33.31**/**0.918**	$\underline{32.05}$/**0.901**	**31.27**/**0.924**
Bicubic	3	30.39/0.868	27.55/0.774	27.21/0.739	24.46/0.735
SRCNN [6]	3	32.75/0.909	29.3/0.822	28.41/0.786	26.24/0.799
VDSR [7]	3	33.67/0.921	29.78/0.832	28.83/0.799	27.14/0.830
LapSRN [20]	3	33.82/0.922	29.87/0.832	28.82/0.798	27.07/0.828
DRRN [11]	3	34.03/0.924	29.96/0.835	$\underline{28.95}$/0.800	**27.53**/$\underline{0.837}$
MS-LapSRN [21]	3	34.01/0.924	29.96/$\underline{0.836}$	28.92/$\underline{0.801}$	27.39/0.835
IDN [22]	3	$\underline{34.11}$/$\underline{0.925}$	$\underline{29.99}$/0.835	$\underline{28.95}$/$\underline{0.801}$	27.42/0.836
LGCN (Ours)	3	**34.12**/**0.926**	**30.01**/**0.837**	**28.96**/**0.803**	$\underline{27.47}$/**0.840**
Bicubic	4	28.42/0.810	26.00/0.703	25.96/0.668	23.14/0.658
SRCNN [6]	4	30.48/0.863	27.52/0.753	26.91/0.712	24.53/0.724
VDSR [7]	4	31.35/0.882	28.02/0.769	27.29/0.726	25.18/0.753
LapSRN [20]	4	31.54/0.885	28.19/0.772	27.32/0.727	25.21/0.756
DRRN [11]	4	31.68/0.888	28.21/0.772	27.38/0.728	25.44/0.764
MS-LapSRN [21]	4	31.74/0.888	28.25/$\underline{0.773}$	**27.42**/$\underline{0.731}$	$\underline{25.45}$/$\underline{0.765}$
IDN [22]	4	**31.82**/$\underline{0.890}$	$\underline{28.25}$/$\underline{0.773}$	$\underline{27.41}$/0.730	25.41/0.763
LGCN (Ours)	4	$\underline{31.79}$/**0.891**	**28.30**/**0.780**	**27.42**/**0.735**	**25.49**/**0.766**

convolutional layer in the memory unit and the 1 $\times$ 1 convolutional layer for hierarchical feature fusion. We use different kernel size for transposed convolutional layer depending on the scaling factor, i.e., the 4 $\times$ 4 transposed convolutional layer with striding 2 for scaling factor 2 $\times$, 5 $\times$ 5 transposed convolutional layer with striding 3 for scaling factor 3 $\times$. To reduce the parameters of reconstruction layer when scaling factor become large, we use two 4 $\times$ 4 convolutional layers with striding 2 for scaling factor 4 $\times$. The first convolutional layer in F-Net has $C_0 = 128$ filters, and the other convolutional layers has $C = 64$ filters, except for the final layer and the dimensionality reduction layer of the channel attention unit. The dimensionality reduction layer of the channel attention unit has $C/8 = 8$ filters, where the reduction ratio r = 8. The final convolutional layer has 1 output channel because we only process the luminance channel.

4.2. Comparisons against the state-of-the-arts

We compare our LGCN with several state-of-the-art lightweight SISR methods, including Bicubic, SRCNN [6], VDSR [7], LapSRN [20], MS-LapSRN [21], DRRN [11] and IDN [22]. We evaluate the super-resolution images with average peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) values [39] on four benchmark datasets. As shown in Table 1, the proposed LGCN method achieves the best performance on most datasets under most scaling factors. Our model performs slightly inferior PSNR on Set5 comparing with IDN [22] on the 2 $\times$ and 4 $\times$ scales. It should be noticed that, the Set5 dataset is the smallest dataset and only contains 5 images. Take the 4 $\times$ scale for example, our model gets 0.05 dB higher than IDN [22] on Set14 dataset, and 0.08 dB improvement against the IDN [22] on the challenging Urban 100 dataset. In addition, our LGCN achieves the highest SSIM values on all datasets for all scaling factors, and can produce the images that have high structural similarity with the original high-resolution image.

We also choose two non-lightweight SISR methods, i.e., SRResNet [40] and EDSR [9] for comparison. The results are reported in Table 2. We can observe that both methods outperform our LGCN. This is a reasonable result since they have deeper and wider network structure that contains large quantities of convolutional layers and parameters. Actually, the parameters of SRResNet [40] and EDSR [9] are 1543K and 43000K, while ours is only 660k.

We also show visual comparisons on Set14, BSD100 and Urban100 for 4 $\times$ SR in Figs. 6–8, respectively. In Fig. 6, we can observe that most of the comparative methods would suffer from blurring artifacts for the flower in red box due to the loss of high frequency information. In contrast, the proposed LGCN method recovers the red dot on the flower clearly. As shown in Fig. 7, LGCN can not only produce clearer horizontal lines on dock with less fake information, but also generate clearer bulge at the side of the ship in the red box in Fig. 7. In Fig. 8, we can observe that LGCN yields sharper curves than the others. These comparisons demonstrate that LGCN can accurately reconstruct details such as the points, the horizontal lines and the curves.

Table 2
Comparison with non-lightweight methods.

	Scale	EDSR [9]	SRResNet [40]	LGCN(Ours)
Set5	2	38.11/0.960	37.85/0.959	37.75/0.960
	3	34.65/0.928	34.25/0.925	34.12/0.926
	4	32.46/0.896	32.00/0.891	31.79/0.891
Set14	2	33.92/0.919	33.55/0.918	33.31/0.918
	3	30.25/0.846	30.27/0.839	30.01/0.837
	4	28.80/0.787	28.50/0.779	28.30/0.780
BSD100	2	32.32/0.901	32.08/0.900	32.05/0.901
	3	29.25/0.809	28.97/0.803	28.96/0.803
	4	27.71/0.742	27.49/0.736	27.42/0.735
Urban100	2	32.93/0.935	31.90/0.929	31.27/0.924
	3	28.80/0.865	28.00/0.847	27.47/0.840
	4	26.64/0.803	26.05/0.781	25.49/0.766
	Paremeters	43000K	1543K	660K

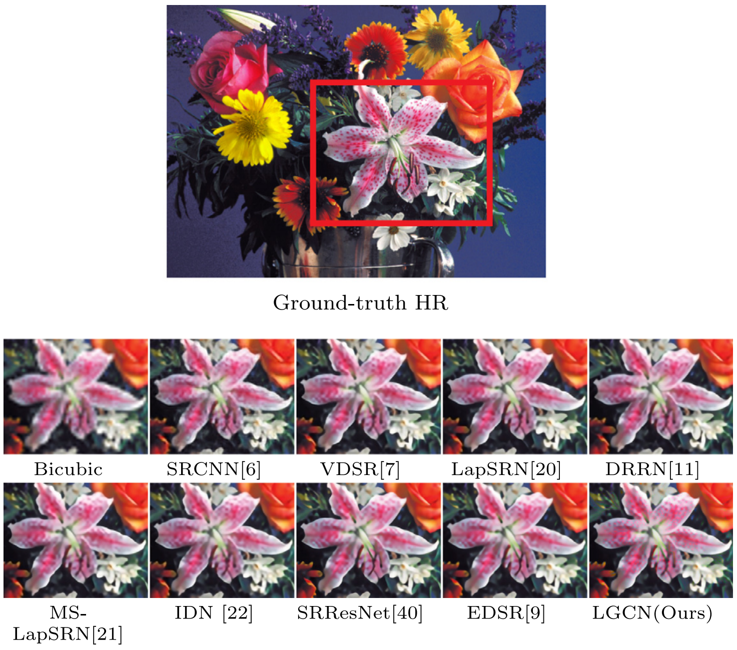


Fig. 6. Visual comparison for 4 × SR on Set14 dataset.

4.3. Computational cost analysis

We use the publicly released codes of the state-of-the-art methods to evaluate the runtime on the same machine with 3.2GHz Intel i7 CPU (16G RAM) and NVIDIA TITAN XP GPU (12G memory). Since the code of SRCNN for testing is based on CPU implementation, we reproduce this model with PyTorch to measure its running time on GPU. We take Set14 under scale 4 × for example. As shown in Fig. 2, our LGCN costs 0.01 second per image, which is as fast as IDN [22], but faster than the other competitors. Specifically, LGCN is approximately 4 times faster than LapSRN [20], 5 times faster than MS-LapSRN [21], more than 40 times faster than SRResNet [40] and EDSR [9]. On the other hand, comparing with the pre-upsampled methods, LGCN is 6 times faster than SRCNN [6], more than 10 times faster than VDSR [7] and DRRN [11].

The proposed LGCN is the fastest algorithm since it applies convolution on LR images, which is different from the SRCNN [6], VDSR [7] and DRRN [11]. These algorithms upsample the image before feeding it to the network, which increases the computational cost. The DRRN [11] and MS-LapSRN [21] adopt the recursive structure to build a very deep network. However, with the increase of the network depth, the running time will increase obviously. And the LapSRN also need deal with the upsampled image when the scaling factor 4 × , thus it will also take more time.

4.4. Ablation studies

In this section, we evaluate the effectiveness of the important components of LGCN via ablation studies. For simplicity, the ablation experiments are performed on the Set5 dataset with scaling factor 4 × .

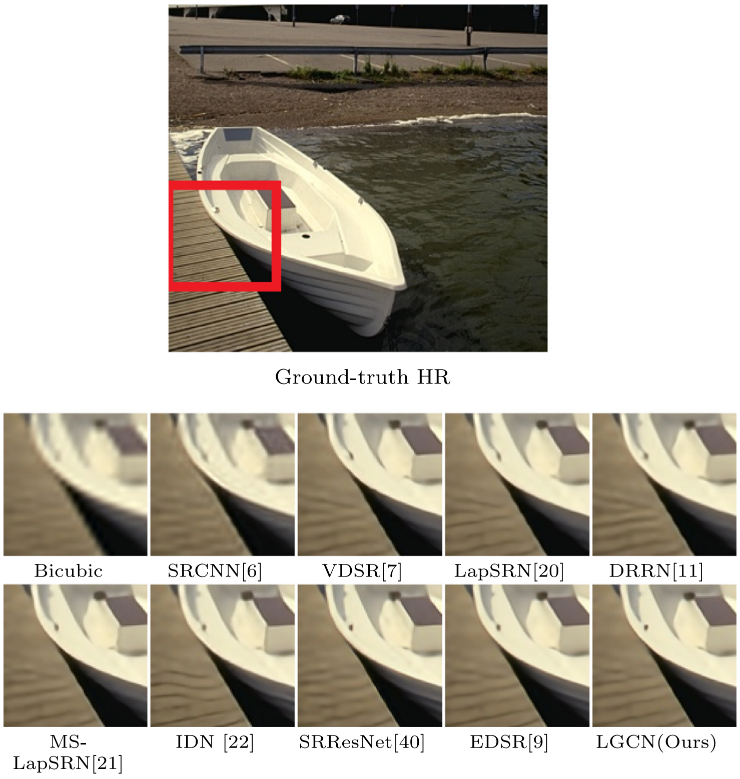


Fig. 7. Visual comparison for 4 × SR on BSD100 dataset.

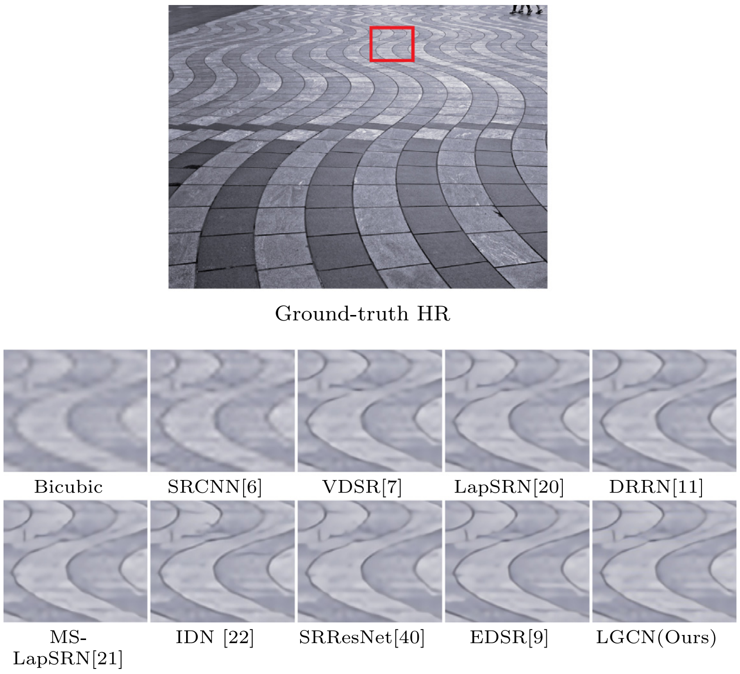


Fig. 8. Visual comparison for 4 × SR on Urban100 dataset.

Table 3
Ablation study of 1 × 1 convolution and densely connected structure in memory unit on Set5 for 4 × SR.

Memory unit				
	1×1 convolution		√	√
	dense connection			√
PSNR(dB)		31.52	31.57	**31.79**

Table 4
Effects of the Channel Attention (CA) unit on Set5 for 4 × SR.

	With CA	Without CA
Parameter	653K	650K
PSNR(dB)	**31.79**	31.66

Effectiveness of the memory unit. Table 2 shows the ablation study of 1 × 1 convolution and densely connected structure. The three models have the same MGCN number of 3. The first column is the baseline, which contains four simply stacked group convolutional layers with plain structure, without 1 × 1 convolution and densely connected structure. It should be noticed that our densely connected structure utilizes 1 × 1 convolution to gather the information from previous layers, the second row of “dense connection” means the unit contains both 1 × 1 convolution and densely connected structure. As shown in Table 2, the baseline achieves a PSNR of 31.52 dB. When only adding 1 × 1 convolution to the baseline, the performance gets 0.05 dB improvement against the baseline. This comparison shows that the 1 × 1 convolutional layer can create a linear combination of the output of group convolutional layer to improve representation ability of the LGCN. This is because outputs from a certain group only relate to the inputs within the group when feature maps was convolved by group convolution, this property blocks information flow between channel groups. By adding 1 × 1 convolutional layer, the information between different groups can be fused. However, as discussed in ResNet [16], the network would get deeper by simply adding 1 × 1 convolution, which has a risk of vanishing gradients. Densely connected structure is capable of alleviating the vanishing gradients by strengthening the flow of information in the unit. Thus, after further applying densely connected structure, the performance is significantly improved to be 31.79 dB. In detail, densely connected structure makes information transmit by multiple shortcuts and thus mitigates the vanishing gradient problem. Moreover, this structure gathers all features before present layers, which helps the information to propagate to higher layers. Therefore, the memory unit leads to significant performance improvement.

Effectiveness of channel attention unit. To demonstrate the effect of channel attention unit, we remove it from the memory group convolutional network, while keeping other parts remained. The quantitative results in Table 3 shows that network with channel attention unit would perform better than those without it (31.79 dB vs. 31.66 dB) with a small increase of parameters (653K vs. 650K). The comparisons firmly indicate the effectiveness of the channel attention unit, which is owned to introducing the channel-wise attention mechanism to make the network focus on more informative features.

Analysis of the group size. We train the proposed LGCN with different group size, i.e., G = 1, 2, and 4 at each block and show their parameters and PSNR in Table 4. We refer G = 1 the standard convolution without groups. When group size is 2, the performance drops down in 0.11 dB against with 222K parameters reduction. When group size is changed to 4, the performance is 0.16 dB lower than that with G = 2 with 110k parameters reduction. It can be seen that with the increase of group number, the performance get worse while the parameter number is decreased rapidly. As a tradeoff, we choose G = 2 in our final LGCN model. As observed in Fig. 9, our LGCN with G = 2 achieves a quite lightweight model as well as performing excellent reconstruction quality. The reason is that each filter in each group is convolved with only half the previous layer’s feature maps, each filter is exactly half the number of parameters of the equivalent normal convolutional layer. By doing so, group convolution significantly reduces computational cost. The DRRN [11], MS-LapSRN [21] require less parameters than LGCN because they adopt the recursive structure to build deeper networks. However, this design will lead to a higher computational burden, whose computational times are 5–10 times than our LGCN model. The SRCNN has the lowest parameters since it only contains 3 convolutional layers. However, its performance is unsatisfying. On the other hand, we achieve a slight performance of 0.05 dB improvement comparing to IDN [22], It can also find that the parameters of SRResNet [40] and EDSR [9] are quite larger than ours, which are 2 times and more than 60 times of ours, respectively. As a whole, LGCN achieves competitive performance and parameters comparing to the state-of-the-art methods.

Analysis of the number of MGCN. We train LGCN with different number of MGCN, i.e., K = 2, 3, and 4 to evaluate the impacts of MGCN number. Table 5 shows the parameter and performance with different K. As we can observe, stacking more convolutional layer will lead to better performance. Thus, when we stack more MGCN, we will get better PSNR. However, it will also bring more computation cost. For better balance between the parameters and performance, we choose $K = 3$ in our final LGCN.

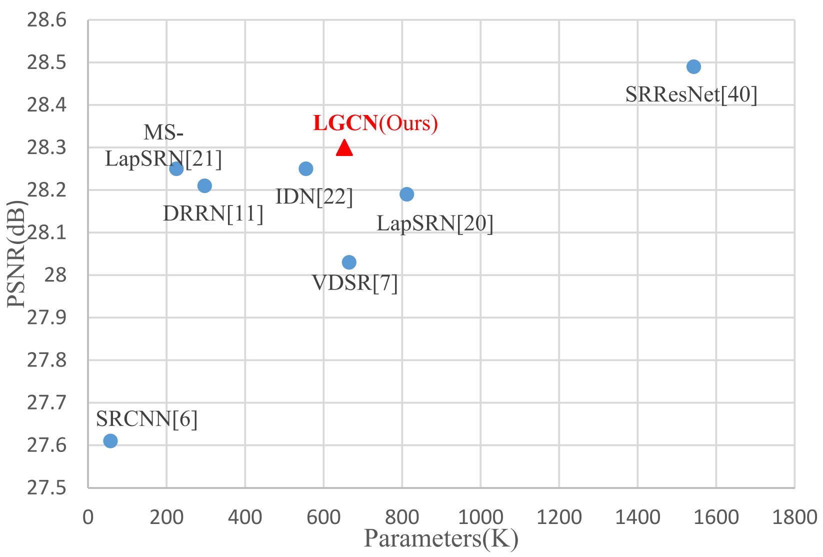


Fig. 9. Model parameters comparison. The results are evaluated on Set14 with the scaling factor 4 × . For clearer comparison, the parameter of EDSR [9] is not shown in the figure, whose parameter is up to 43,000K.

Table 5
Evaluation on Group size.

Group Size	1	2	4
Parameter	875K	653K	543K
PSNR(dB)	31.90	**31.79**	31.63

Table 6
Evaluation on the number of MGCN.

K	2	3	4
Parameter	515K	653K	792K
PSNR(dB)	31.66	**31.79**	31.89

5. Conclusion

In this work, a lightweight group convolutional network called LGCN is proposed for fast and accurate SISR. In the memory group convolutional network, we applied memory mechanism by utilizing group convolutional layer and 1 × 1 convolutional layer with densely connected structure to efficiently learn multi-level representations. We also presented a novel channel attention unit to model the interdependencies among channels. Furthermore, we cascaded several memory group convolutional network and adopted hierarchical feature fusion to combine all-level feature to boost model performance. Our LGCN model is evaluated on four benchmark datasets and achieves competing performance against the state-of-the-art methods on both accuracy and parameter amount.

Declaration of Competing Interest

Authors declare that they have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled, ”Lightweight Group Convolutional Network for Single Image Super-Resolutio”.

CRediT authorship contribution statement

Aiping Yang: Conceptualization, Investigation, Resources. **Bingwang Yang:** Data curation, Methodology, Software. **Zhong Ji:** Investigation, Methodology, Writing - original draft. **Yanwei Pang:** Supervision, Visualization, Writing - review & editing. **Ling Shao:** Validation, Writing - review & editing.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (Grants No. 61771329 and 61632018). The authors are very grateful for NVIDIA's support in providing GPUs that made this work possible.

References

[1] H. Liu, Z. Fu, J. Han, L. Shao, S. Hou, Y. Chu, Single image super-resolution using multi-scale deep encoder–decoder with phase congruency edge map guidance, Inf. Sci. (Ny) 473 (2019) 44–58.
[2] Y. Zhao, R. Wang, W. Jia, J. Yang, W. Wang, W. Gao, Local patch encoding-based method for single image super-resolution, Inf. Sci. (Ny) 433 (2018) 292–305.
[3] W.W.W. Zou, P.C. Yuen, Very low resolution face recognition problem, IEEE Trans. Image Process. 21 (1) (2012) 327–340.
[4] P. Lu, L. Barazzetti, V. Chandran, K. Gavaghan, S. Weber, N. Gerber, M. Reyes, Highly accurate facial nerve segmentation refinement from CBCT/CT imaging using a super-resolution classification approach, IEEE Trans. Biomed. Eng. 65 (1) (2018) 178–188.
[5] M.S. Sajjadi, B. Schölkopf, M. Hirsch, Enhancenet: Single image super-resolution through automated texture synthesis, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4501–4510.
[6] C. Dong, C.C. Loy, K. He, X. Tang, Image super-resolution using deep convolutional networks, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2) (2016) 295–307.
[7] J. Kim, J. Kwon Lee, K. Mu Lee, Accurate image super-resolution using very deep convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1646–1654.
[8] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[9] B. Lim, S. Son, H. Kim, S. Nah, K.M. Lee, Enhanced deep residual networks for single image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshops, 2017, pp. 1132–1140.
[10] J. Kim, J. Kwon Lee, K. Mu Lee, Deeply-recursive convolutional network for image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1637–1645.
[11] Y. Tai, J. Yang, X. Liu, Image super-resolution via deep recursive residual network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2790–2798.
[12] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2261–2269.
[13] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.
[14] T. Tong, G. Li, X. Liu, Q. Gao, Image super-resolution using dense skip connections, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4799–4807.
[15] X.J. Mao, C. Shen, Y.B. Yang, Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 2802–2810.
[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[17] M. Haris, G. Shakhnarovich, N. Ukita, Deep backprojection networks for super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1664–1673.
[18] C. Dong, C.C. Loy, X. Tang, Accelerating the super-resolution convolutional neural network, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 391–407.
[19] 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 IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874–1883.
[20] W.-S. Lai, J.-B. Huang, N. Ahuja, M.-H. Yang, Deep laplacian pyramid networks for fast and accurate super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5835–5843.
[21] W.-S. Lai, J.-B. Huang, N. Ahuja, M.-H. Yang, Fast and accurate image super-resolution with deep laplacian pyramid networks, IEEE Trans. Pattern Anal. Mach. Intell. (2018) 2599–2613.
[22] Z. Hui, X. Wang, X. Gao, Fast and accurate single image super-resolution via information distillation network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 723–731.
[23] N. Ahn, B. Kang, K. Sohn, Fast, accurate, and lightweight super-resolution with cascading residual network, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 256–272.
[24] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, Squeezenet: alexnet-level accuracy with 50x fewer parameters and < 0.5 Mb model size, International Conference on Learning Representations (2017) 1–13.
[25] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5987–5995.
[26] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: efficient convolutional neural networks for mobile vision applications, arXiv:1704.04861 (2017).
[27] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.
[28] Y. Tai, J. Yang, X. Liu, C. Xu, Memnet: A persistent memory network for image restoration, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4539–4547.
[29] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the International Conference on Machine Learning, 2010, pp. 807–814.
[30] 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.
[31] J. Yang, J. Wright, T.S. Huang, Y. Ma, Image super-resolution via sparse representation, IEEE Trans. Image Process. 19 (11) (2010) 2861–2873.
[32] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2011) 898–916.
[33] M. Bevilacqua, A. Roumy, C. Guillemot, M. line Alberi Morel, Low-complexity single-image super-resolution based on nonnegative neighbor embedding, in: Proceedings of the British Machine Vision Conference, 2012, pp. 135.1–135.10.
[34] 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, pp. 711–730.
[35] 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 IEEE International Conference on Computer Vision, 2, 2001, pp. 416–423.
[36] J.-B. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5197–5206.
[37] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, International Conference on Learning Representations (2015) 1–15.

[38] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1026–1034.
[39] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[40] 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 IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4681–4690.