Fast, Accurate, and Lightweight Super-Resolution with Cascading Residual Network

Namhyuk Ahn[0000−0003−1990−9516], Byungkon Kang[0000−0001−8541−2861], and Kyung-Ah Sohn[0000−0001−8941−1188]

Department of Computer Engineering,
Ajou University
{aa0dfg,byungkon,kasohn}@ajou.ac.kr

Abstract. In recent years, deep learning methods have been successfully applied to single-image super-resolution tasks. Despite their great performances, deep learning methods cannot be easily applied to real-world applications due to the requirement of heavy computation. In this paper, we address this issue by proposing an accurate and lightweight deep network for image super-resolution. In detail, we design an architecture that implements a *cascading mechanism* upon a residual network. We also present variant models of the proposed cascading residual network to further improve efficiency. Our extensive experiments show that even with much fewer parameters and operations, our models achieve performance comparable to that of state-of-the-art methods.

Keywords: Super-Resolution, Deep Convolutional Neural Network

1 Introduction

Super-resolution (SR) is a computer vision task that reconstructs a high-resolution (HR) image from a low-resolution (LR) image. Specifically, we are concerned with single image super-resolution (SISR), which performs SR using a single LR image. SISR is generally difficult to achieve due to the fact that computing the HR image from an LR image is a many-to-one mapping. Despite such difficulty, SISR is a very active area because it can offer the promise of overcoming resolution limitations, and could be used in a variety of applications such as video streaming or surveillance system.

Recently, convolutional neural network-based (CNN-based) methods have provided outstanding performance in SISR tasks [6,20,24]. From the SRCNN [6] that has three convolutional layers to MDSR [26] that has more than 160 layers, the depth of the network and the overall performance have dramatically grown over time. However, even though deep learning methods increase the quality of the SR images, they are not suitable for real-world scenarios. From this point of view, it is important to design lightweight deep learning models that are practical for real-world applications. One way to build a lean model is reducing the number

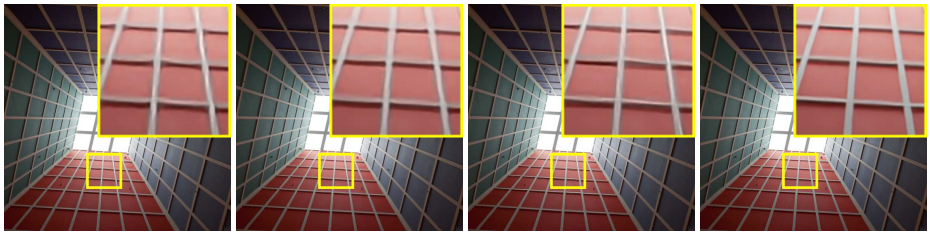


Fig. 1: Super-resolution result of our methods compared with existing methods.

of parameters. There are many ways to achieve this [11, 19], but the most simple and effective approach is to use a *recursive network*. For example, DRCN [21] uses a recursive network to reduce redundant parameters, and DRRN [35] improves DRCN by adding a residual architecture to it. These models decrease the number of model parameters effectively when compared to the standard CNN and show good performance. However, there are two downsides to these models: 1) They first upsample the input image before feeding it to the CNN model, and 2) they increase the depth or the width of the network to compensate for the loss due to using a recursive network. These points enable the model to maintain the details of the image when reconstructed, but at the expense of the increased number of operations and inference time.

Most of the works that aim to build a lean model focused primarily on reducing the number of parameters. However, as mentioned above, the number of operations is also an important factor to consider in real-world scenarios. Consider a situation where an SR system operates on a mobile device. Then, the execution speed of the system is also of crucial importance from a user-experience perspective. Especially the battery capacity, which is heavily dependent on the amount of computation performed, becomes a major problem. In this respect, reducing the number of operations in the deep learning architectures is a challenging and necessary step that has largely been ignored until now. Another scenario relates to applying SR methods to video streaming services. The demand for streaming media has skyrocketed and hence requires large storage to store massive multimedia data. It is therefore imperative to compress data using lossy compression techniques before storing. Then, an SR technique can be applied to restore the data to the original resolution. However, because latency is the most critical factor in streaming services, the decompression process (i.e., super-resolution) has to be performed in real-time. To do so, it is essential to make the SR methods lightweight in terms of the number of operations.

To handle these requirements and improve the recent models, we propose a Cascading residual network (CARN) and its variant CARN-Mobile (CARN-M). We first build our CARN model to increase the performance and extend it to CARN-M to optimize it for speed and the number of operations. Following the FSRCNN [7], CARN family take the LR images and compute the HR

counterparts as the output of the network. The middle parts of our models are designed based on the ResNet [13]. The ResNet architecture has been widely used in deep learning-based SR methods [26, 35] because of the ease of training and superior performance. In addition to the ResNet architecture, CARN uses a *cascading mechanism* at both the local and the global level to incorporate the features from multiple layers. This has the effect of reflecting various levels of input representations in order to receive more information. In addition to the CARN model, we also provide the CARN-M model that allows the designer to tune the trade-off between the performance and the *heaviness* of the model. It does so by means of the efficient residual block (residual-E) and recursive network architecture, which we describe in more detail in Section 3.

In summary, our main contributions are as follows: **1)** We propose CARN, a neural network based on the cascading modules, which achieves high performance on SR task (Fig. 1). Our cascading modules, effectively boost the performance via multi-level representation and multiple shortcut connections. **2)** We also propose CARN-M for efficient SR by combining the efficient residual block and the recursive network scheme. **3)** We show through extensive experiments, that our model uses only a modest number of operations and parameters to achieve competitive results. Our CARN-M, which is the more lightweight SR model, shows comparable results to others with much fewer operations.

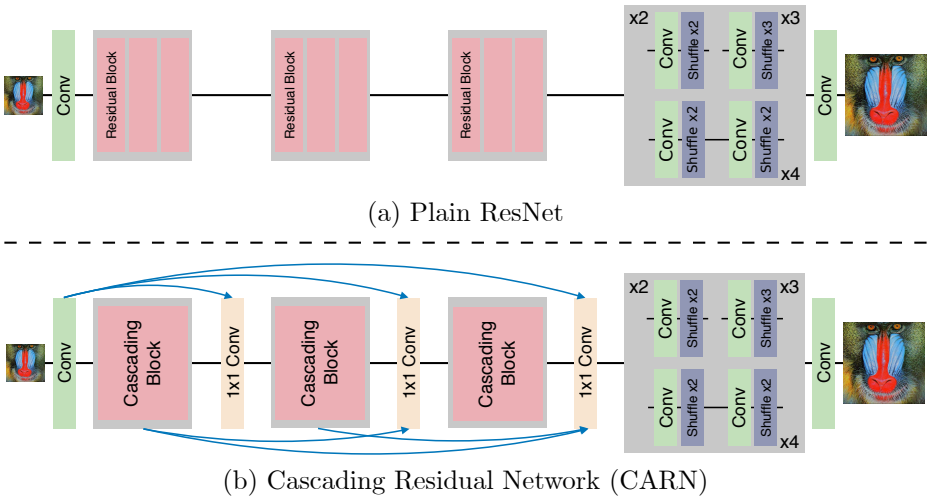
2 Related Work

Since the success of AlexNet [23] in image recognition task [5], many deep learning approaches have been applied to diverse computer vision tasks [9, 27, 30, 40]. The SISR task is one such task, and we present an overview of the deep learning-based SISR in section 2.1. Another area we deal with in this paper is model compression. Recent deep learning models focus on squeezing model parameters and operations for application in low-power computing devices, which has many practical benefits in real-world applications. We briefly review in section 2.2.

2.1 Deep Learning Based Image Super-Resolution

Recently, deep learning based models have shown dramatic improvements in the SISR task. Dong et al. [6] first proposed a deep learning-based SR method, SRCNN, which outperformed traditional algorithms. However, SRCNN has a large number of operations compared to its depth, since network takes upsampled images as input. Taking a different approach from SRCNN, FSRCNN [7] and ESPCN [33] upsample images at the end of the networks. By doing so, it leads to the reduction in the number of operations compared to the SRCNN. However, the overall performance could be degraded if there are not enough layers after the upsampling process. Moreover, they cannot manage multi-scale training, as the input image size differs for each upsampling scale.

Despite the fact that the power of deep learning comes from *deep* layers, the aforementioned methods have settled for shallow layers because of the difficulty



(a) Plain ResNet

(b) Cascading Residual Network (CARN)

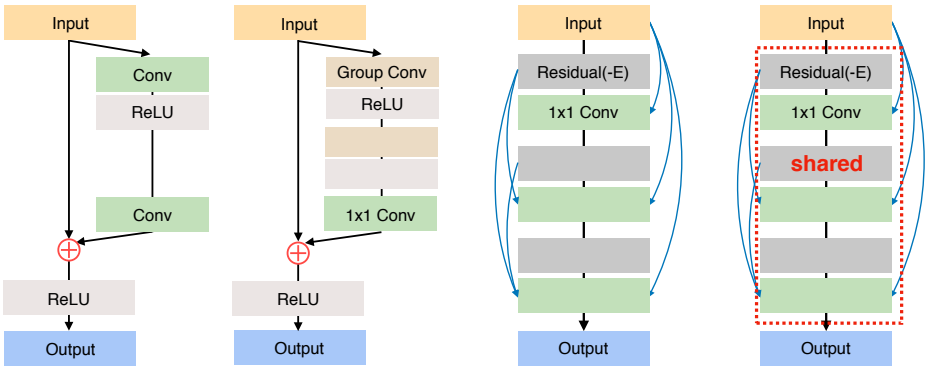
Fig. 2: Network architectures of plain ResNet (**top**) and the proposed CARN (**bottom**). Both models are given an LR image and upsample to HR at the end of the network. In the CARN model, each residual block is changed to a cascading block. The blue arrows indicate the global cascading connection.

in training. To better harness the depth of deep learning models, Kim et al. [20] proposed VDSR, which uses *residual learning* to map the LR images $\mathbf{x}$ to their residual images $\mathbf{r}$. Then, VDSR produces the SR images $\mathbf{y}$ by adding the residual back into the original, *i.e.*, $\mathbf{y} = \mathbf{x} + \mathbf{r}$. On the other hand, LapSRN [24] uses a Laplacian pyramid architecture to increase the image size gradually. By doing so, LapSRN effectively performs SR on extremely low-resolution cases with a fewer number of operations compared to VDSR.

Another issue of deep learning-based SR is how to reduce the parameters and operation. For example, DRCN [21] uses a recursive network to reduce parameters by engaging in redundant usages of a small number of parameters. DRRN [35] improves DRCN by combining the recursive and residual network schemes to achieve better performance with fewer parameters. However, DRCN and DRRN use very deep networks to compensate for the loss of performance and hence these require heavy computing resources. Hence, we aim to build a model that is lightweight in both size and computation. We will briefly discuss previous works that address such model efficiency issues in the following section.

2.2 Efficient Neural Network

There has been rising interest in building small and efficient neural networks [11, 16, 19]. These approaches can be categorized into two groups: **1)** Compressing pretrained networks, and **2)** designing small but efficient models. Han et al. [11] proposed deep compressing techniques, which consist of pruning, vector quanti-



(a) Residual Block (b) Residual-E Block (c) Cascading Block (d) Recursive Block

Fig. 3: Simplified structures of (a) residual block (b) efficient residual block (residual-E), (c) cascading block and (d) recursive cascading block. The $\oplus$ operations in (a) and (b) are element-wise addition for residual learning.

zation, and Huffman coding to reduce the size of a pretrained network. In the latter category, SqueezeNet [19] builds an AlexNet-based architecture and achieves comparable performance level with 50× fewer parameters. MobileNet [16] builds an efficient network by applying depthwise separable convolution introduced in Sifre et al. [34]. Because of this simplicity, we also apply this technique in the residual block with some modification to achieve a lean neural network.

3 Proposed Method

As mentioned in Section 1, we propose two main models: CARN and CARN-M. CARN is designed to be a high-performing SR model while suppressing the number of operations compared to the state-of-the-art methods. Based on CARN, we design CARN-M, which is a much more efficient SR model in terms of both parameters and operations.

3.1 Cascading Residual Network

Our CARN model is based on ResNet [14]. The main difference between CARN and ResNet is the presence of local and global cascading modules. Fig. 2 (b) graphically depicts how the global cascading occurs. The outputs of intermediary layers are *cascaded* into the higher layers, and finally converge on a single 1x1 convolution layer. Note that the intermediary layers are implemented as cascading blocks, which host local cascading connections themselves. Such local cascading operations are shown in Figure 2(c) and (d). Local cascading is almost identical to a global one, except that the unit blocks are plain residual blocks.

To express the implementation formally, let f be a convolution function and τ be an activation function. Then, we can define the i-th residual block R_i, which

has two convolutions followed by a residual addition, as

$$R_i(H^{i-1}; W_R^i) = \tau(f(\tau(f(H^{i-1}; W_R^{i,1})); W_R^{i,2}) + H^{i-1}). \tag{1}$$

Here, H^i is the output of the i-th residual block, W_R^i is the parameter set of the residual block, and $W_R^{i,j}$ is the parameter of the j-th convolution layer in the i-th block. With this notation, we denote the output feature of the final residual block of ResNet as H^u, which becomes the input to the upsampling block.

$$H^u = R_u\left(\dots\left(R_1\left(f\left(\boldsymbol{X}; W_c\right); W_R^1\right)\right)\dots; W_R^u\right). \tag{2}$$

Note that because our model has a single convolution layer before each residual block, the first residual block gets $f(\boldsymbol{X}; W_c)$ as input, where W_c is the parameter of the convolution layer.

In contrast to ResNet, our CARN model has a local cascading block illustrated in block (c) of Fig. 3 instead of a plain residual block. In here, we denote $B^{i,j}$ as the output of the j-th residual block in the i-th cascading block, and W_c^i as the set of parameters of the i-th local cascading block. Then, the i-th local cascading block B_{local}^i is defined as

$$B_{local}^i\left(H^{i-1}; W_l^i\right) \equiv B^{i,U}, \tag{3}$$

where $B^{i,U}$ is defined recursively from the $B^{i,u}$'s as:

$$\begin{aligned} B^{i,0} &= H^{i-1} \\ B^{i,u} &= f\left(\left[I, B^{i,0}, \dots, B^{i,u-1}, R^u\left(B^{i,u-1}; W_R^u\right)\right]; W_c^{i,u}\right) \quad \text{for } u = 1, \dots, U. \end{aligned}$$

Finally, we can define the output feature of the final cascading block H^b by combining both the local and global cascading. Here, H^0 is the output of the first convolution layer. And we we fix $u = b = 3$ for our CARN and CARN-M.

$$\begin{aligned} H^0 &= f\left(\boldsymbol{X}; W_c\right) \\ H^b &= f\left(\left[H^0, \dots, H^{b-1}, B_{local}^u\left(H^{b-1}; W_B^b\right)\right]\right) \quad \text{for } b = 1, \dots, B. \end{aligned} \tag{4}$$

The main difference between CARN and ResNet lies in the cascading mechanism. As shown in Fig. 2, CARN has global cascading connections represented as the blue arrows, each of which is followed by a 1×1 convolution layer. Cascading on both the local and global levels has two advantages: **1)** The model incorporates features from multiple layers, which allows learning multi-level representations. **2)** Multi-level cascading connection behaves as multi-level shortcut connections that quickly propagate information from lower to higher layers (and vice-versa, in case of back-propagation).

CARN adopts a multi-level representation scheme as in [25,28], but we apply this arrangement to a variety of feature levels to boost performance, as shown in equation 4. By doing so, our model reconstructs the LR image based on multi-level features. This facilitates the model to restore the details and contexts of the image simultaneously. As a result, our models effectively improve not only primitive objects but also complex objects.

Another reason for adopting the cascading scheme is two-fold: First, the propagation of information follows multiple paths [17, 32]. Second, by adding extra convolution layers, our model can learn to choose the right pathway with the given input information flows. However, the strength of multiple shortcuts is degraded when we use only one of local or global cascading, especially the local connection. We elaborate the details and present a case study on the effects of cascading mechanism in Section 4.4.

3.2 Efficient Cascading Residual Network

To improve the efficiency of CARN, we propose an efficient residual (residual-E) block. We use a similar approach to the MobileNet [16], but use group convolution instead of depthwise convolution. Our residual-E block consists of two 3×3 group and one pointwise convolution, as shown in Fig. 3 (b). The advantage of using group convolution over the depthwise convolution is that it makes the efficiency of the model tunable. The user can choose the group size appropriately since the group size and performance are in a trade-off relationship. The analysis on the cost efficiency of using the residual-E block is as follows.

Let K be the kernel size and C_{in}, C_{out} be the number of input and output channels. Because we retain the feature resolution of the input and output by padding, we can denote F to be both the input and output feature size. Then, the cost of a plain residual block is as $2 \times (K \cdot K \cdot C_{in} \cdot C_{out} \cdot F \cdot F)$. Note that we only count the cost of convolution layers and ignore the addition or activation because both the plain and the efficient residual blocks have the same amount of cost in terms of addition and activation.

Let G be the group size. Then, the cost of a residual-E block, which consist of two group convolutions and one pointwise convolution, is as given in equation 5.

$$2 \times \left(K \cdot K \cdot C_{in} \cdot \frac{C_{out}}{G} \cdot F \cdot F \right) + C_{in} \cdot C_{out} \cdot F \cdot F \tag{5}$$

By changing the plain residual block to our efficient residual block, we can reduce the computation by the ratio of

$$\frac{2 \times \left(K \cdot K \cdot C_{in} \cdot \frac{C_{out}}{G} \cdot F \cdot F \right) + C_{in} \cdot C_{out} \cdot F \cdot F}{2 \times (K \cdot K \cdot C_{in} \cdot C_{out} \cdot F \cdot F)} = \frac{1}{G} + \frac{1}{2K^2}. \tag{6}$$

Because our model uses a kernel of size 3×3 for all group convolutions, and the number of channels is constantly 64, using an efficient residual block instead of a standard residual block can reduce the computation from 1.8 up to 14 times depending on the group size. To find the best trade-off between performance and computation, we performed an extensive case study in Section 4.4.

To further reduce the parameters, we apply a technique similar to the one used by the recursive network. That is, we make the parameters of the Cascading blocks shared, effectively making the blocks recursive. Fig. 3 (d) shows our block after applying the recursive scheme. This approach reduces the parameters by up to three times of their original number.

3.3 Comparison to Recent Models

Comparison to SRDenseNet. SRDenseNet [37] uses dense block and skip connection. The differences from our model are: **1)** We use global cascading, which is more general than the skip connection. In SRDenseNet, all levels of features are combined at the end of the final dense block, but our global cascading scheme connects all blocks, which behaves as multi-level skip connection. **2)** SRDenseNet preserves local information of dense block via concatenation operations, while we gather it progressively by 1×1 convolution layers. The use of additional 1×1 convolution layers results in a higher representation power.
Comparison to MemNet. The motivation of MemNet [36] and ours is similar. However, there are two main differences from our mechanism. **1)** Inside of the memory blocks of MemNet, the output features of each recursive units are concatenated at the end of the network and then fused with 1×1 convolution. On the other hand, we fuse the features at every possible point in the local block, which can boost up the representation power via the additional convolution layers and nonlinearity. In general, this representation power is often not met because of the difficulty of training. However, we overcome this problem by using both local and global cascading mechanisms. We will discuss the details on Section 4.4. **2)** MemNet takes upsampled images as input so the number of multi-adds is larger than ours. The input to our model is a LR image and we upsample it at the end of the network in order to achieve computational efficiency.

4 Experimental Results

4.1 Datasets

There exist diverse single image super-resolution datasets, but the most widely used ones are the 291 image set by Yang et al. [39] and the Berkeley Segmentation Dataset [2]. However, because these two do not have sufficient images for training a deep neural network, we additionally use the DIV2K dataset [1]. The DIV2K dataset is a newly-proposed high-quality image dataset, which consists of 800 training images, 100 validation images, and 100 test images. Because of the richness of this dataset, recent SR models [4, 8, 26, 31] use DIV2K as well. We use the standard benchmark datasets such as Set5 [3], Set14 [39], B100 [29] and Urban100 [18] for testing and benchmarking.

4.2 Implementation and Training Details

We use the RGB input patches of size 64×64 from the LR images for training. We sample the LR patches randomly and augment them with random horizontal flips and 90 degree rotation. We train our models with the ADAM optimizer [22] by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ in 6×10^5 steps. The minibatch size is 64, and the learning rate begins with 10^{-4} and is halved every 4×10^5 steps. All the weights and biases are initialized by $\theta \sim U(-k,\ k)$ with $k = 1/\sqrt{c_{in}}$ where, c_{in} is the number of channels of input feature map.

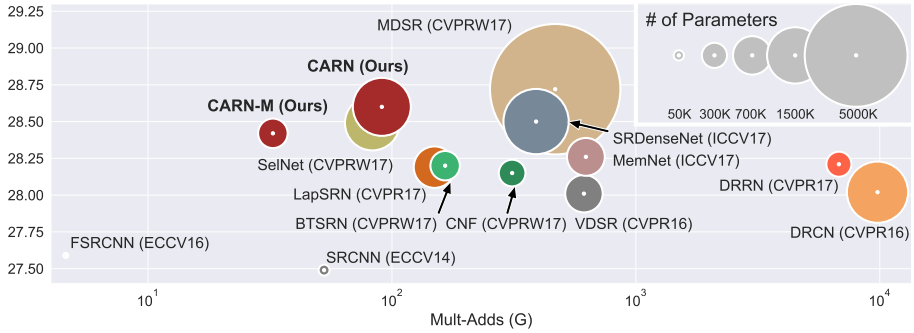


Fig. 4: Trade-off between performance vs. number of operations and parameters on Set14 $\times 4$ dataset. The x-axis and the y-axis denote the Multi-Adds and PSNR, and the size of the circle represents the number of parameters. The Mult-Adds is computed by assuming that the resolution of HR image is 720p.

The most well-known and effective weight initialization methods are given by Glorot et al. [10] and He et al. [12]. However, such initialization routines tend to set the weights of our multiple narrow 1×1 convolution layers very high, resulting in an unstable training. Therefore, we sample the initial values from a uniform distribution to alleviate the initialization problem.

To train our model in a multi-scale manner, we first set the scaling factor to one of $\times 2$, $\times 3$, and $\times 4$ because our model can only process a single scale for each batch. Then, we construct and argument our input batch, as described above. We use the L1 loss as our loss function instead of the L2. The L2 loss is widely used in the image restoration task due to its relationship with the peak signal-to-noise ratio (PSNR). However, in our experiments, L1 provides better convergence and performance. The downside of the L1 loss is that the convergence speed is relatively slower than that of L2 without the residual block. However, this drawback could be mitigated by using a ResNet style model.

4.3 Comparison with State-of-the-art Methods

We compare the proposed CARN and CARN-M with state-of-the-art SR methods on two commonly-used image quality metrics: PSNR and the structural similarity index (SSIM) [38]. One thing to note here is that we represent the number of operations by Mult-Adds. Mult-Adds is the number of composite multiply-accumulate operations for a single image. We assume the HR image size to be 720p (1280×720) to calculate Multi-Adds. In Fig. 4, we compare our CARN family against the various benchmark algorithms in terms of the Mult-Adds and the number of the parameters on the Set14 $\times 4$ dataset. Here, our CARN model outperforms all state-of-the-art models that have less than 5M parameters. Especially, CARN has the similar number of parameters to that of DRCN [21], SelNet [4] and SRDenseNet [37], but we outperform all three models.

The MDSR [26] achieves better performance than ours, which is not surprising because MDSR has 8M parameters which are nearly six times more parameters than ours. The CARN-M model also outperforms most of the benchmark methods and shows comparable results against the heavy models.

Moreover, our models are most efficient in terms of the computation cost: CARN shows second best results with 90.9G Mult-Adds, which is on par with SelNet [4]. This efficiency mainly comes from the *late-upsample* approach that many recent models [7, 24, 37] used. In addition, our novel cascading mechanism shows increased performance compared to the models with the same manner. For example, CARN outperforms SelNet by a margin of 0.11 PSNR using almost identical computation resources. Also, the CARN-M model obtains comparable results against computationally-expensive models, while only requiring the similar number of the operations with respect to SRCNN.

Table 1 also shows the quantitative comparisons of the performances over the benchmark datasets. Note that MDSR is excluded from this table, because we only compare models that have roughly similar number of parameters as ours; MDSR has a parameter set whose size is four times larger than that of the second-largest model. Our CARN exceeds all the previous methods on numerous benchmark dataset. CARN-M model achieves comparable results using very few operations. We would also like to emphasize that although CARN-M has more parameters than SRCNN or DRRN, it is tolerable in real-world scenarios. The sizes of SRCNN and CARN-M are 200KB and 1.6MB, respectively, all of which are acceptable on recent mobile devices.

To make our models even more lightweight, we apply the multi-scale learning approach. The benefit of using multi-scale learning is that it can process multiple scales using a single trained model. This helps us alleviate the burden of heavy-weight model size when deploying the SR application on mobile devices; CARN(-M) only needs a single fixed model for multiple scales, whereas even the state-of-the-art algorithms require to train separate models for each supported scale. This property is well-suited for real-world products because the size of the applications has to be fixed while the scale of given LR images could vary. Using the multi-scale learning to our models increases the number of parameters, since the network has to contain possible upsampling layers. On the other hand, VDSR and DRRN do not require this extra burden, even if multi-scale learning is performed, because they upsample the image before processing it.

In Fig. 6, we visually illustrate the qualitative comparisons over three datasets (Set14, B100 and Urban100) for $\times 4$ scale. It can be seen that our model works better than others and accurately reconstructs not only stripes and line patterns, but also complex objects such as hand and street lamps.

4.4 Model Analysis

To further investigate the performance behavior of the proposed methods, we analyze our models via ablation study. First, we show how local and global cascading modules affect the performance of CARN. Next, we analyze the trade-off between performance vs. parameters and operations.

Table 1: Quantitative results of deep learning-based SR algorithms. Red/blue text: best/second-best.

Scale	Model	Params	MultAdds	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM
2	SRCNN [6]	57K	52.7G	36.66/0.9542	32.42/0.9063	31.36/0.8879	29.50/0.8946
	FSRCNN [7]	12K	6.0G	37.00/0.9558	32.63/0.9088	31.53/0.8920	29.88/0.9020
	VDSR [20]	665K	612.6G	37.53/0.9587	33.03/0.9124	31.90/0.8960	30.76/0.9140
	DRCN [21]	1,774K	9,788.7G	37.63/0.9588	33.04/0.9118	31.85/0.8942	30.75/0.9133
	CNF [31]	337K	311.0G	37.66/0.9590	33.38/0.9136	31.91/0.8962	-
	LapSRN [24]	813K	29.9G	37.52/0.9590	33.08/0.9130	31.80/0.8950	30.41/0.9100
	DRRN [35]	297K	6,796.9G	37.74/0.9591	33.23/0.9136	32.05/0.8973	31.23/0.9188
	BTSRN [8]	410K	207.7G	37.75/-	33.20/-	32.05/-	31.63/-
	MemNet [36]	677K	623.9G	37.78/0.9597	33.28/0.9142	32.08/0.8978	31.31/0.9195
	SelNet [4]	974K	225.7G	37.89/0.9598	33.61/0.9160	32.08/0.8984	-
	CARN (ours)	1,592K	222.8G	37.76/0.9590	33.52/0.9166	32.09/0.8978	31.51/0.9312
	CARN-M (ours)	412K	91.2G	37.53/0.9583	33.26/0.9141	31.92/0.8960	30.83/0.9233
3	SRCNN [6]	57K	52.7G	32.75/0.9090	29.28/0.8209	28.41/0.7863	26.24/0.7989
	FSRCNN [7]	12K	5.0G	33.16/0.9140	29.43/0.8242	28.53/0.7910	26.43/0.8080
	VDSR [20]	665K	612.6G	33.66/0.9213	29.77/0.8314	28.82/0.7976	27.14/0.8279
	DRCN [21]	1,774K	9,788.7G	33.82/0.9226	29.76/0.8311	28.80/0.7963	27.15/0.8276
	CNF [31]	337K	311.0G	33.74/0.9226	29.90/0.8322	28.82/0.7980	-
	DRRN [35]	297K	6,796.9G	34.03/0.9244	29.96/0.8349	28.95/0.8004	27.53/0.8378
	BTSRN [8]	410K	176.2G	34.03/-	29.90/-	28.97/-	27.75/-
	MemNet [36]	677K	623.9G	34.09/0.9248	30.00/0.8350	28.96/0.8001	27.56/0.8376
	SelNet [4]	1,159K	120.0G	34.27/0.9257	30.30/0.8399	28.97/0.8025	-
	CARN (ours)	1,592K	118.8G	34.29/0.9255	30.29/0.8407	29.06/0.8034	27.38/0.8404
	CARN-M (ours)	412K	46.1G	33.99/0.9236	30.08/0.8367	28.91/0.8000	26.86/0.8263
4	SRCNN [6]	57K	52.7G	30.48/0.8628	27.49/0.7503	26.90/0.7101	24.52/0.7221
	FSRCNN [7]	12K	4.6G	30.71/0.8657	27.59/0.7535	26.98/0.7150	24.62/0.7280
	VDSR [20]	665K	612.6G	31.35/0.8838	28.01/0.7674	27.29/0.7251	25.18/0.7524
	DRCN [21]	1,774K	9,788.7G	31.53/0.8854	28.02/0.7670	27.23/0.7233	25.14/0.7510
	CNF [31]	337K	311.0G	31.55/0.8856	28.15/0.7680	27.32/0.7253	-
	LapSRN [24]	813K	149.4G	31.54/0.8850	28.19/0.7720	27.32/0.7280	25.21/0.7560
	DRRN [35]	297K	6,796.9G	31.68/0.8888	28.21/0.7720	27.38/0.7284	25.44/0.7638
	BTSRN [8]	410K	165.2G	31.85/-	28.20/-	27.47/-	25.74/-
	MemNet [36]	677K	623.9G	31.74/0.8893	28.26/0.7723	27.40/0.7281	25.50/0.7630
	SelNet [4]	1,417K	83.1G	32.00/0.8931	28.49/0.7783	27.44/0.7325	-
	SRDenseNet [37]	2,015K	389.9G	32.02/0.8934	28.50/0.7782	27.53/0.7337	26.05/0.7819
	CARN (ours)	1,592K	90.9G	32.13/0.8937	28.60/0.7806	27.58/0.7349	26.07/0.7837
	CARN-M (ours)	412K	32.5G	31.92/0.8903	28.42/0.7762	27.44/0.7304	25.63/0.7688

Cascading Modules. Table 2 presents the ablation study on the effect of local and global cascading modules. In this table, the baseline is ResNet, CARN-NL is CARN without local cascading and CARN-NG is CARN without global cascading. The network topologies are all same, but because of the 1×1 convolution layer, the overall number of parameters is increased by up to 10%.

We see that the model with only global cascading (CARN-NL) shows better performance than the baseline because the global cascading mechanism effectively carries mid- to high-level frequency signals from shallow to deep layers. Furthermore, by gathering all features before the upsampling layers, the model can better leverage multi-level representations. By incorporating multi-level representations, the CARN model can consider a variety of information from many different receptive fields when reconstructing the image.

Table 2: Effects of the global and local cascading modules measured on the Set14 ×4 dataset. CARN-NL represents CARN without local cascading and CARN-NG without global cascading. CARN is our final model.

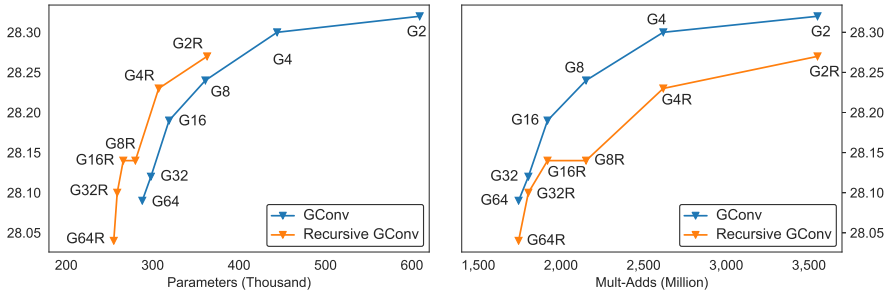
	Baseline	CARN-NL	CARN-NG	CARN
Local Cascading			✓	✓
Global Cascading		✓		✓
# Params.	1,444K	1,481K	1,555K	1,592K
PSNR	28.43	28.45	28.42	**28.52**

Somewhat surprisingly, using only local cascading blocks (CARN-NG) harms the performance. As discussed in He et al. [15], multiplicative manipulations such as 1×1 convolution on the shortcut connection can hamper information propagation, and thus lead to complications during optimization. Similarly, cascading connections in the local cascading blocks of CARN-NG behave as shortcut connections inside the residual blocks. Because these connections consist of concatenation and 1×1 convolutions, it is natural to expect performance degradation. That is, the advantage of multi-level representation is limited to the inside of each local cascading block. Therefore, there appears to be no benefit of using the cascading connection because of the increased number of multiplication operations in the cascading connection. However, CARN uses both local and global cascading levels and outperforms all three models. This is because the global cascading mechanism eases the information propagation issues that CARN-NG suffers from. In detail, information propagates globally via global cascading, and information flows in the local cascading blocks are fused with the ones that come through global connections. By doing so, information is transmitted by multiple shortcuts and thus mitigates the vanishing gradient problem. In other words, the advantage of multi-level representation is leveraged by the global cascading connections, which help the information to propagate to higher layers.

Efficiency Trade-off. Fig. 5 depicts the trade-off study of PSNR vs. parameters, and PSNR vs. operations in relation to the efficient residual block and recursive network. In this experiment, we evaluate all possible group sizes of the efficient residual block for both the recursive and non-recursive cases. In both graphs, the blue line represents the model that does not use the recursive scheme and the orange line is the model that uses recursive cascading block.

Although all efficient models perform worse than the CARN, which shows 28.70 PSNR, the number of parameters and operations are decreased dramatically. For example, the *G64* shows a five-times reduction in both parameters and operations. However, unlike the comparable result that is shown in Howard et al. [16], the degradation of performance is more pronounced in our case.

Next, we observe the case which uses the recursive scheme. As illustrated in Fig. 5b, there is no change in the Mult-Adds but the performance worsens, which



(a) Trade-off of parameters-PSNR (b) Trade-off of operations-PSNR

Fig. 5: Results of using efficient residual block and recursive network in terms of PSNR vs. parameters(**left**) and PSNR vs. operations(**right**). We evaluate all models on Set14 with $\times 4$ scale. *GConv* represents the group size of group convolution and *R* means the model with the recursive network scheme (i.e., *G4R* represents group four with recursive cascading blocks).

seems reasonable given the decreased number of parameters in the recursive scheme. On the other hand, Fig. 5a shows that using the recursive scheme makes the model achieve better performance with fewer parameters. Based on these observations, we decide to choose the group size as four in the efficient residual block and use the recursive network scheme as our CARN-M model. By doing so, CARN-M reduces the number of parameters by five times and the number of operations by nearly four times with a loss of 0.29 PSNR compared to CARN.

5 Conclusion

In this work, we proposed a novel cascading network architecture that can perform SISR accurately and efficiently. The main idea behind our architecture is to add multiple cascading connections starting from each intermediary layer to the others. Such connections are made on both the local (block-wise) and global (layer-wise) levels, which allows for the efficient flow of information and gradient. Our experiments show that employing both types of connections greatly outperforms those using only one or none at all.

We wish to further develop this work by applying our technique to video data. Many streaming services require large storage to provide high-quality videos. In conjunction with our approach, one may devise a service that stores low-quality videos that go through our SR system to produce high-quality videos on-the-fly.

Acknowledgement.

This research was supported through the National Research Foundation of Korea (NRF) funded by the Ministry of Education: NRF-2016R1D1A1B03933875 (K.-A. Sohn) and NRF-2016R1A6A3A11932796 (B. Kang).

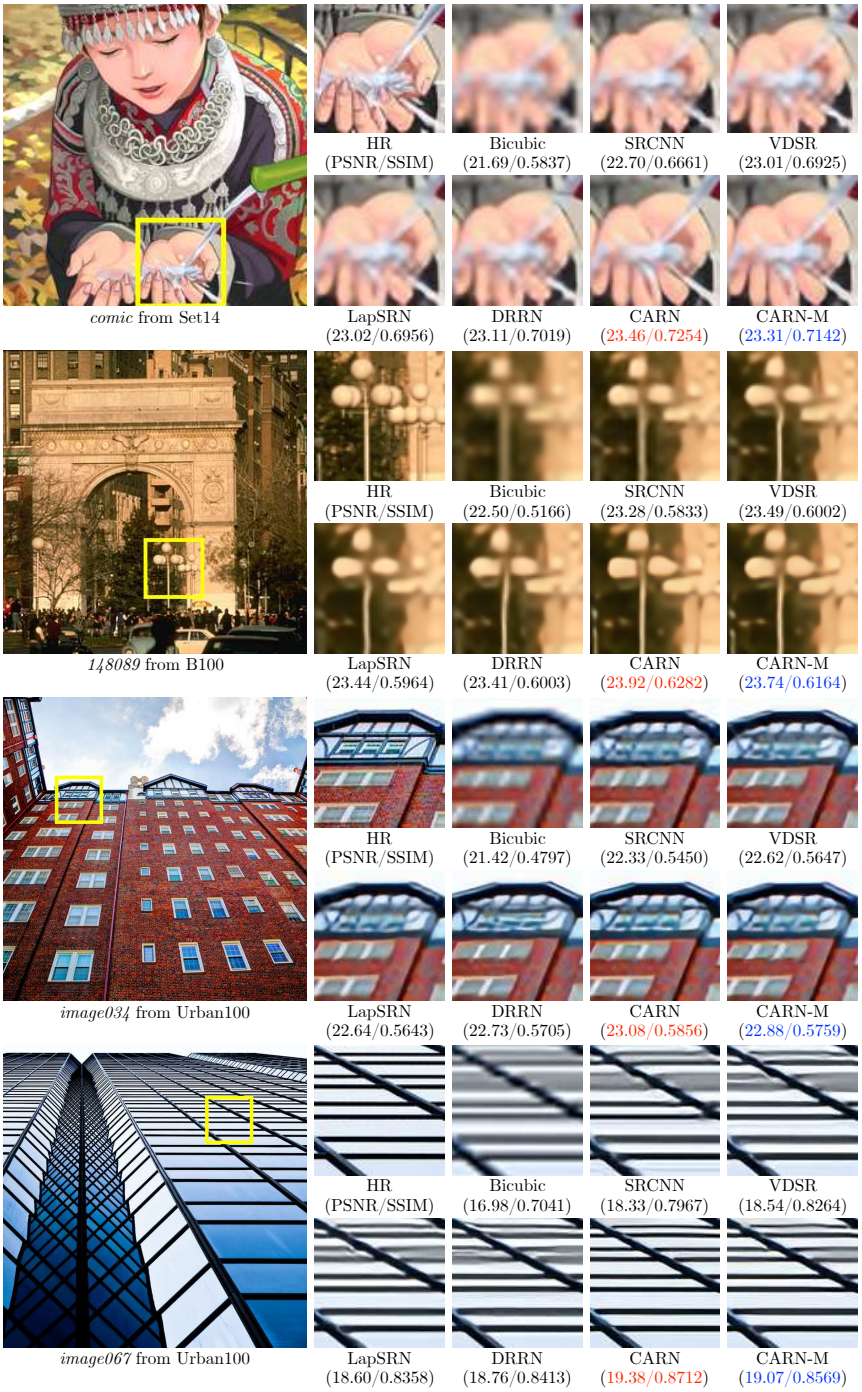


Fig. 6: Visual qualitative comparison on $\times 4$ scale datasets.

References

1. Agustsson, E., Timofte, R.: Ntire 2017 challenge on single image super-resolution: Dataset and study. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)
2. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence **33**(5), 898–916 (2011)
3. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: Proceedings of the British Machine Vision Conference (BMVC) (2012)
4. Choi, J.S., Kim, M.: A deep convolutional neural network with selection units for super-resolution. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
6. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: Proceedings of the European Conference on Computer Vision (ECCV) (2014)
7. Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: Proceedings of the European Conference on Computer Vision (ECCV) (2016)
8. Fan, Y., Shi, H., Yu, J., Liu, D., Han, W., Yu, H., Wang, Z., Wang, X., Huang, T.S.: Balanced two-stage residual networks for image super-resolution. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)
9. Girshick, R.: Fast r-cnn. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015)
10. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the International Conference on Artificial Intelligence and Statistics (2010)
11. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. Proceedings of the International Conference on Learning Representations (ICLR) (2016)
12. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
15. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Proceedings of the European Conference on Computer Vision (ECCV) (2016)
16. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

17. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
18. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
19. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
20. Kim, J., Kwon Lee, J., Mu Lee, K.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
21. Kim, J., Kwon Lee, J., Mu Lee, K.: Deeply-recursive convolutional network for image super-resolution. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. Proceedings of the International Conference on Learning Representations (ICLR) (2015)
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the Conference on Neural Information Processing Systems (NIPS) (2012)
24. Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Deep laplacian pyramid networks for fast and accurate super-resolution. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
25. Lee, J., Nam, J.: Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. IEEE Signal Processing Letters **24**(8), 1208–1212 (2017)
26. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)
27. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: Proceedings of the European Conference on Computer Vision (ECCV) (2016)
28. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
29. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of the International Conference on Computer Vision (ICCV) (2001)
30. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015)
31. Ren, H., El-Khamy, M., Lee, J.: Image super resolution based on fusing multiple convolution neural networks. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)
32. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (2015)
33. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-

pixel convolutional neural network. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
34. Sifre, L., Mallat, S.: Rigid-motion scattering for image classification. Ph.D. thesis, Citeseer (2014)
35. Tai, Y., Yang, J., Liu, X.: Image super-resolution via deep recursive residual network. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
36. Tai, Y., Yang, J., Liu, X., Xu, C.: Memnet: A persistent memory network for image restoration. In: Proceedings of the International Conference on Computer Vision (ICCV) (2017)
37. Tong, T., Li, G., Liu, X., Gao, Q.: Image super-resolution using dense skip connections. In: Proceedings of the International Conference on Computer Vision (ICCV) (2017)
38. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
39. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. IEEE transactions on image processing **19**(11), 2861–2873 (2010)
40. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: Proceedings of the European Conference on Computer Vision (ECCV) (2016)