Efficient Deep Neural Network for Photo-realistic Image Super-Resolution

Namhyuk Ahn, Byungkon Kang, Kyung-Ah Sohn*

Department of Computer Engineering, Ajou University, Suwon-si, Korea

Abstract

Recent progress in the deep learning-based models has improved photo-realistic (or perceptual) single-image super-resolution significantly. However, despite their powerful performance, many models are difficult to apply to the real-world applications because of the heavy computational requirements. To facilitate the use of a deep learning model under such demands, we focus on keeping the model fast and lightweight while maintaining its performance. In detail, we design an architecture that implements a cascading mechanism on a residual network to boost the performance with limited resources via multi-level feature fusion. Moreover, we adopt group convolution and weight-tying for our proposed model in order to achieve extreme efficiency. In addition to our network, we use the adversarial learning paradigm and a multi-scale discriminator approach. By doing so, we show that the performances of the proposed models surpass those of the recent methods, which have a complexity similar to ours, for both traditional pixel-based and perception-based tasks. To verify the effectiveness of our models, we investigate through extensive internal experiments and benchmark using various datasets.

Keywords: Super-resolution, Photo-realistic, Convolutional Neural Network, Efficient Model

1. Introduction

Image super-resolution (SR) is a longstanding computer vision task that can be widely used in numerous applications. It focuses on recovering a high-resolution (HR) image from low-resolution (LR) images. In particular, single-image super-resolution (SISR) performs SR using a single LR image. Since the SISR problem is a many-to-one mapping, constructing an effective SISR algorithm is a challenging task. In spite of the difficulties, SISR has been actively studied because it can potentially be applied to a variety of services. Recently, deep learning-based methods have shown prominent performance on this task [1, 2, 3]. The major trend of these algorithms is not only stacking more layers to their models [2, 4] but also designing and assembling internal blocks and network topologies [5, 6, 7] to achieve more accurate results.

Even though SR performance continues to improve, there still exists a gap between the quantitative scores and human-perceived judgment. Various methods adopt pixel-based (or distortion-based) subjective error functions, such as mean squared error (MSE) or L1 loss, to train the SR networks. Minimizing such objectives leads to high peak signal-to-noise ratio (PSNR) score, which is a commonly used quantity measure in the SR community. However, the ability to restore the high-frequency details in such cases is limited, since pixel-based error functions only capture the difference between two images pixel-wise. Moreover, they often result in blurry output image, thus usually disagreeing with the subjective evaluation scores given by human judges. To address such shortcomings, several deep learning-based methods optimize their networks in a perceptual manner to improve human-visual quality. Starting with SRGAN [8], most of the models that aim for good perceptual quality employ the generative adversarial network (GAN) [9] paradigm and perceptual loss [10]. ESRGAN [11] achieves the best perceptual quality by considering generator and discriminator simultaneously.

Although deep learning-based algorithms increase the quality of the SR images significantly, using these models for the real-world scenarios is another challenge. There are many cases that require not only performance but also efficiency, such as streaming services or mobile platform-based applications. However, the recently-proposed EPSR [12] and ESRGAN [11] use more than 50 convolutional layers, which can be computationally heavy on mobile devices. From this point of view, it is important to design a lightweight model that can handle such demands.

Several works make efforts to design a *lean* SR model by reducing the number of parameters. One of the most simple and effective approaches is to construct the network in a recursive manner. However, even though models with recursive structures [5] show good performance and efficiency in terms of the parameters, they have two

*Corresponding author.
Email address: kasohn@ajou.ac.kr (Kyung-Ah Sohn)

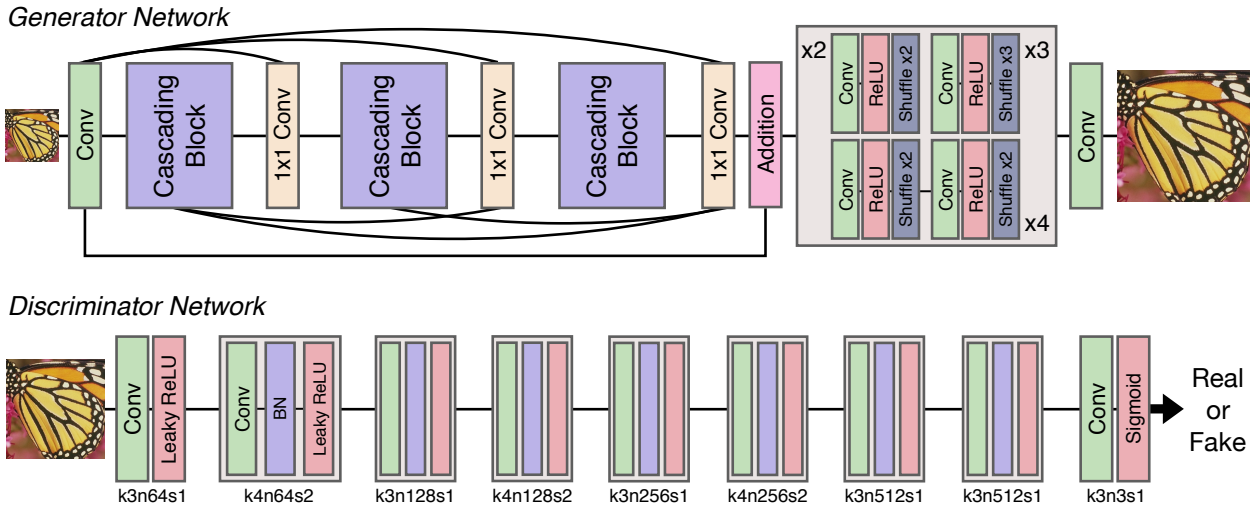


Figure 1: **Network architecture.** (top) Generator network. This network consists of cascading blocks and upsample blocks. (bottom) Discriminator network with corresponding kernel size (k), number of feature map (n) and stride (s) indicated for each convolution layer.

downsides: **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 performance loss caused by the use of a recursive network. These characteristics enable the model to maintain the details of the images when reconstructed, but at the expense of the increased number of operations and inference time.

However, as mentioned earlier, the number of operations is also an important factor to consider in real-world demands. For the SR systems that operate on mobile devices, the execution speed also plays an important role 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 is a challenging and necessary step that has largely been ignored until now. A relevant practical scenario can be found in video streaming services. The demand for streaming media has skyrocketed, and hence large storage for massive multimedia data are required. 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 such services, the decompression process has to be performed in near-real-time. To do so, it is essential to make the SR methods lightweight in terms of the number of operations in order to satisfy timing constraints.

To handle these requirements and improve the recent models, we propose a photo-realistic cascading residual network (PCARN), which is an extended version of our preliminary work [13]. Following the ESPCN [14], the PCARN takes the LR images and computes the HR counterparts as the output of the network. Based on this architecture, we introduce a cascading mechanism at both the local and the global levels to incorporate the features from multiple layers. It has the effect of reflecting various levels of input representations in order to receive more information that can be used in the restoration process. Besides, we adopt the adversarial training to our cascading residual network. To do that, we set the cascading residual network as generator and add discriminator network which distinguish whether the input images are from HR or SR set as shown in Fig. 1. Additionally, we also enhance discriminator to make the model generating images with high perception using a multi-scale discriminator strategy rather than using standard discriminator. The multi-scale discriminator consists of multiple networks, where each network is in charge of handling a certain scale. It makes the generator and discriminator improve the ability to preserve the details by taking into account both the coarse and fine textures. Furthermore, as in our prior work [13], we build the PCARN-M (mobile) to allow the users to tune the trade-off between the performance and the heaviness of the model. It is implemented by using the efficient residual block (EResidual) and recursive network scheme.

In summary, our contributions are as follows: **1)** We propose PCARN, a neural network model based on novel cascading modules that can effectively boost the performance via multi-level representation and multiple shortcut connections. **2)** With GAN-based learning and a multi-scale discriminator, our model can capture fine details effectively. **3)** We also propose PCARN-M for efficient SR by combining the efficient residual block and the recursive network scheme. Experimental results demonstrate that our method is substantially faster and more lightweight than the recent deep learning-based methods for both distortion- and perceptual-based tasks.

2. Related Work

Single-image super-resolution has been extensively studied in the literature. Here, we first focus on the deep learning-based SR models in Sec. 2.1. In Sec. 2.2, we

discuss recent studies to make a photo-realistic super-resolution model that has attracted increasing interest recently. Finally, we briefly review the model compression approaches in Sec. 2.3.

2.1. Deep Learning-based SR

Recently, the performance of the SR has been greatly improved with powerful capabilities of the deep learning-based methods. As a pioneer work, Dong et al. [1] propose a deep learning-based model SRCNN that works much better than the traditional algorithms. However, SRCNN requires large computation resources compared to its depth, since the model takes upsampled images as input. On the other hand, FSRCNN [15] and ESPCN [14] take LR images as inputs and upsample the output at the end of the network. This strategy reduces the computation substantially compared to the *early-upsample* scheme. However, the performance could be degraded since most of the recovering processes are done in the LR space. Another issue is that it is tricky to apply the multi-scale training technique [2] because of the resolution-mismatch problem between the input images across the different scale factors.

An additional shortcoming of the aforementioned methods is that they only use a few convolutional layers because of training instability. To tackle this issue, VDSR [2] introduces global residual learning and shows significant improvement over the previous methods by using 20 convolutional layers. The global residual learning maps the LR image $\mathbf{x}$ to its residual image $\mathbf{r}$. Then, it produces the SR image $\mathbf{y}$ by adding the residual back into the original, *i.e.*, $\mathbf{y} = \mathbf{x} + \mathbf{r}$. This paradigm facilitates training a deep model with fast and stable convergence. Another approach is progressive upsampling [3, 16, 17], which upsamples the intermediary features periodically to restore the image details gradually. By doing so, those methods effectively perform SR on extremely low-resolution cases compared to the one-stage upsampling manner.

One issue of applying a deep learning-based method is the efficiency of the model. That is, there is a problematic increase in the size of the model. To address this concern, most of the previous studies aim to build a lightweight model in terms of the number of parameters. For example, DRCN [5] uses a recursive network to reduce the number of parameters so that the model training is much easier even with small-sized data. Similarly, MemNet [18] uses recursive units in the memory block to boost the performance with only a small number of parameters. This idea is applied to the progressive model as well: The MSLapSRN [19] improves LapSRN [3] by tying the parameters of each feature-embedding blocks and results in performance superior to that of the LapSRN.

However, many of the parameter-efficient methods use very deep networks to compensate for degraded performance caused by the use of the recursive scheme and thus require heavy computing resources. On the other hand, we aim to build a model that is lightweight in both size and computational aspect.

2.2. Photo-realistic SR

Generally, deep learning-based SR models are trained using pixel-based (or distortion-based) loss functions (*e.g.,* MSE or L1 loss). The network with these objectives can be optimized easily, but it tends to create blurry artifacts and fails to recover the details such as object edges. This characteristic can be problematic since a human can judge the absence of high-frequency information effortlessly [8]. Hence, to overcome the inherent issue of using distortion-based losses, a generative adversarial network [9] has been adopted to the SR field [8] recently. By doing so, GAN-based methods show promising results in preserving human-perceptive quality. However, since using only an adversarial loss makes the training process unstable, most of the GAN-based models are trained with the addition of pixel losses. To overcome the inherent problems of pixel losses, Johnson et al. [10] introduces the perceptual loss that calculates the distance between the embedded features of two images.

Besides the SRGAN, many recent works [20, 21, 11] try to improve the perceptual quality. The ENet [20] and TSRN [21] adopt texture-matching loss [22] in combination with an adversarial training and perceptual losses. Texture-matching loss was originally used in the texture synthesis problem. Given a target texture image, it generates the output image by iteratively matching the statistics extracted from a pretrained network to the target texture. By using texture information in the SR problem, the model can produce more realistic textures and reduce artifacts. On the other hand, ESRGAN [11] improves the SRGAN in a different direction. Internally, ESRGAN replaces standard residual units with the residual-in-residual dense block (RRDB) inspired by the SRDenseNet [6] and RDN [7]. Then, it adds the relative discriminator loss [23] to increase the visual quality. However, perceptually-oriented models are not suitable for real world applications despite their high performances. On the contrary, our proposed models create photo-realistic images with a reasonable amount of computation in order to suit the real-world demands.

In photo-realistic SR task, measuring the quality of the resulting image is a major issue. There are many studies that propose distortion-based metrics for the image quality assessment [24]. But these works do not always reflect the visual quality, and some metrics often contradict human judgment [8]. For example, Blau and Michaeli [25] study the trade-off between the average distortion and perceptual quality. Considering the trade-off, we mainly use NIMA [26] and LPIPS [27] perceptual quality metrics as our benchmark test. NIMA is a newly proposed metric that predicts the distribution of human opinion scores using a deep learning model. It makes the assessment non-referencially, where all the evaluation is done without ground-truth images. The LPIPS measures the perceptual quality by using the distance between two deep features generated by the pretrained network. Upon the pretrained network, they add an extra linear layer and fine-tune it to

the human perceptual dataset. In our experiments, we use a fine-tuned AlexNet (linear version 0.1) [28] as the pretrained network needed to compute these measures..

2.3. Efficient Neural Network

There has been rising interest in building a small and efficient network [29, 30, 31]. These approaches can be categorized into three groups: **1)** Compressing pretrained networks using pruning or quantizing techniques, **2)** transferring knowledge of a deep model to a shallow one, and **3)** designing small but efficient models. In this section, we summarize the latter category, which aims to build a lean neural network in terms of design engineering, as it matches our approach most closely.

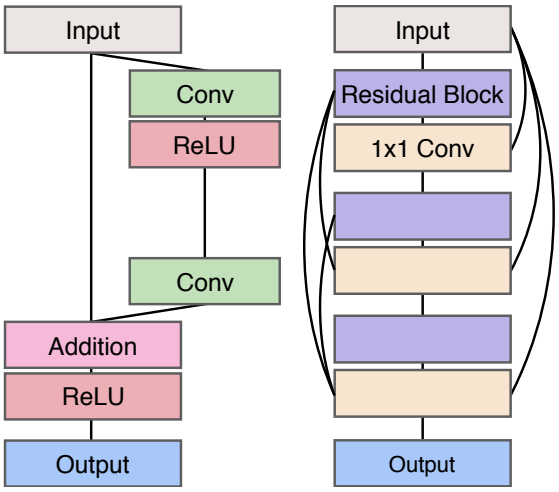
Iandola et al. [32] introduces SqueezeNet to build a parameter-efficient architecture based on the AlexNet [28]. By doing so, they achieve comparable performance level with 50× fewer parameters than the baseline model. Unlike the SqueezeNet, MobileNet [30] aims to decrease the number of operations in order to reduce the inference runtime. This model decomposes the standard convolution to 1×1 and depthwise separable convolutions used in the previous studies [33]. While the MobileNet cuts down the computational cost effectively, 1×1 convolution becomes the new bottleneck, and thus can be the limitation to pushing down the overall cost. To mitigate this issue, ShuffleNet and its variant [34, 35] use the channel shuffle unit following the 1×1 group convolution.
Referring to the recent literature, we apply a depthwise separable convolution technique in the residual block with a generalized form to achieve a fast and lightweight SR model.

3. Our Methods

Our proposed photo-realistic cascading residual network (PCARN) is built on our prior model, CARN [13]. In Sec. 3.1, we introduce the generator of our PCARN. Then, we describe the mechanism extending the photo-realistic SR in Sec. 3.2. Finally, PCARN-M, mobile version of PCARN is shown in Sec. 3.3.

3.1. Cascading Residual Network

The generator network of our proposed PCARN is based on CARN, which is in turn based on the ResNet [36]. The prime difference between ResNet and our model is the presence of local and global cascading modules. Fig. 1 (top) graphically depicts how the global cascading occurs. The outputs of intermediary features are cascaded into the higher blocks and finally converge on a single 1×1 convolution layer. Note that the intermediary modules are implemented as cascading blocks, which also host cascading connections themselves in a local way. Such local cascading operations are shown in Figure 2, which shows that the local connection is almost identical to a global one, except that the backbone units are the residual blocks.



(a) Residual Block (b) Cascading Block

Figure 2: **Structures of local blocks.** (a) Residual block. (b) Cascading block composed of residual blocks and local cascading connections.

To express how cascading works formally, let f be a convolution layer and τ be a nonlinearity, ReLU in our case. Now we can define the i-th residual block R_i, which has two convolutions followed by an additive operation, as

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

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 these notations, we denote the output feature of the final residual block of ResNet as H^u and it becomes the input to the upsampling block. Note that since our model has an entry convolution layer, the first residual block takes $f(\boldsymbol{X}; W_c)$ as input, where W_c is the parameter of the convolution layer.

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

As mentioned above, we replace the standard residual block with the local cascading block that is shown in Fig. 2 (b). To formulate the local cascading as well, 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 in Equation 3.

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

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

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

Finally, we define the output feature of the final cascading block H^b by combining both the local and the global cascading with H^0, the output of the entry layer.

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

On top of our preliminary work [13], we make some modifications to the model to boost up the performance.

First, inspired by the VDSR [2], we attach global residual learning in our framework. To do that, we aggregate the output of the entry layers and the final 1×1 convolution layer right before the upsampling block. Formally, it can be written as

$$O = H^b + H^0, \tag{5}$$

where the final feature map O becomes the input to the upsampling block. The effect of this final addition might appear redundant, since the output of the first convolution is already added to the 1×1 before being added again in the next step. Nonetheless, we found that this duplicate addition is beneficial to the overall performance with little computational overhead. Second, unlike the CARN, we adjust the positions of the nonlinearities in the network. That is, we eliminate the nonlinearities following the 1×1 convolution layer. Additionally, we attach additional nonlinearities in the upsampling unit in order to increase the expressive power of the network.

By applying the cascading mechanism on the local and global levels, we can get two advantages: **1)** The model incorporates features from multiple layers, which allows learning a multi-level representation. **2)** The multi-level cascading connection operates as a multi-level shortcut connection that easily propagates information from lower to higher layers (and vice-versa, in the case of back-propagation). Hence, the model can reconstruct the LR image based on multi-level features, and the upsampling unit also upsamples images by taking diverse features into account. Thus, our design helps the model to restore the details and contexts of the image.

3.2. Photo-realistic Cascading Residual Network

Following Goodfellow et al. [9], we define a discriminator network D, which we optimize in an alternative procedure along with the PCARN network G. Using the discriminator and generator, we denote the adversarial loss as:

$$\begin{aligned} L_{GAN}(G, D) = \mathbb{E}_{I_{HR}}\left[\log D(I_{HR})\right] + \\ \mathbb{E}_{I_{LR}}\left[\log(1 - D(G(I_{LR})))\right], \end{aligned} \tag{6}$$

where I_{HR} and I_{LR} denote the HR and LR images, respectively. The idea of adversarial loss is that it trains the generative model G to fool the discriminator D, whereas the discriminator is trained to distinguish whether the images are from the SR or the HR sets. This formulation encourages the generator to create perceptually superior images compared to the pixel-based (distortion-based) losses. This is contrary to the distortion-based SR solutions, which are obtained by minimizing pixel-wise error metrics, such as the MSE.

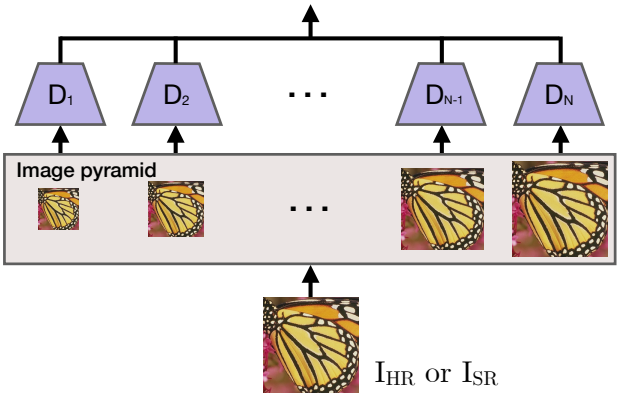


Figure 3: **Conceptual illustration of the multi-scale discriminator.** Input image is downsampled using average pooling and each image is taken by the corresponding scale discriminator. Total discriminative loss is calculated by summing over all the scale losses.

Many previous works have mixed the adversarial loss with a traditional pixel-based loss to stabilize the training process In this case, the task of a generator is not only to fool the discriminator but also to create an SR image similar to the HR. We also take this option but use the VGG loss instead of the pixel-based loss to avoid the blurring artifact. The VGG loss [10] is the distance between the outputs of the ReLU layer of the pre-trained VGGNet [37]. Formally, we denote the output feature map of the j-th ReLU following convolutional layer before the i-th pooling layer as $\phi_{i,j}$. Then, we define the VGG loss as the L2 distance between the feature representation of the HR image I_{HR}, and the super-resoluted image $G(I_{LR})$:

$$\begin{aligned} L_{VGG}(G) = \frac{1}{W_{i,j} H_{i,j}} \sum_{x}^{W_{i,j}} \sum_{y}^{H_{i,j}} \left[\phi_{i,j}(I_{HR})_{x,y}\right. \\ \left. -\phi_{i,j}(G(I_{LR}))_{x,y}\right]^2. \end{aligned} \tag{7}$$

Here, $W_{i,j}$ and $H_{i,j}$ are the spatial resolutions of the feature map. In our work, we use $i = j = 5$.

To enhance the fine details of the computed outputs, we adopt the multi-scale discriminator [38] strategy as depicted in Fig. 3. The main idea is to use multiple discriminators instead of a single one to make each discriminator handle a specific scale. Thus, it allows the model to gather information across coarse- to fine-resolution images. To do so, we first downsample the input image (SR or HR) to make an image pyramid. Then, the scaled images are fed into the corresponding discriminators and finally the multi-scale discriminator loss L_D^{MS} is calculated by collecting each of the losses as in the equation below.

$$L_D^{MS} = \sum_i^S D_i(F_i(\mathbf{I})), \tag{8}$$

where $\mathbf{I}$ is the input image and $F(.)$ is the scale-specific downsample function. In all our experiments, we use average pooling as the downsampling module and set S as three.

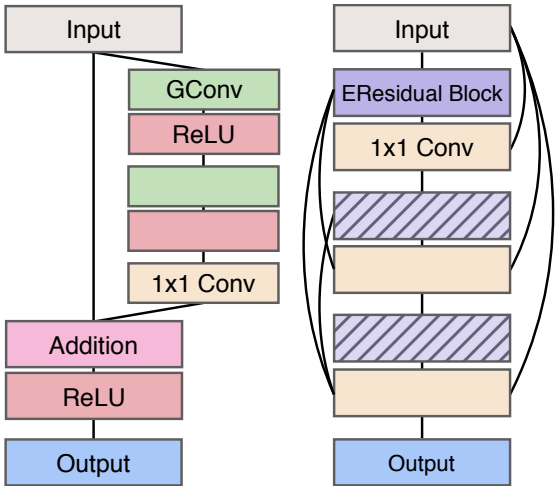
The total loss for the generator is computed by summing the multi-scale GAN loss and VGG loss as:

$$L_G = L_{GAN}^{MS} + \lambda L_{VGG}, \tag{9}$$

where L_{GAN}^{MS} denotes the adversarial loss in terms of the generator with multi-scale discriminator and λ is the coefficients to balance different loss terms.

3.3. Efficient Photo-realistic Cascading Residual Network

To improve the efficiency of PCARN, we propose an efficient residual and cascading block of the generator. This approach is analogous to the MobileNet [30], but we use group convolution instead of depthwise separable convolution. Our efficient residual (EResidual) block is composed of two consecutive 3×3 group convolutions and a single pointwise convolution, as shown in Fig. 4 (a). The advantage of using group convolution over the depthwise separable convolution is that it makes the efficiency of the model manually tunable. Thus, the user can choose the appropriate group size for the desired performance, since the group size and the performance are in a trade-off relationship.



(a) EResidual Block (b) ECascading Block

Figure 4: **Simplified structures of efficient cascading blocks.** (a) Efficient residual block, and (b) is the efficient cascading block. Hatched boxes in (b) denote the residual block with a parameter tying.

The analysis of the cost efficiency of using the EResidual block is as follows. Let K be the kernel size and C_{in}, C_{out} be the number of input and output channels. Since we retain the spatial resolution of the feature map by the padding, we can denote F to be both the input and output feature size. Then, the cost of a standard residual block is

$$2 \times \left(K^2 \cdot C_{in} \cdot C_{out} \cdot F^2\right). \tag{10}$$

Note that we exclude the cost of addition or nonlinearity, and consider only the convolution layers. This is because both the standard and the efficient blocks have the same number of such modules and these occupy a negligible portion of the entire computational cost.

Let G be the group size. Then, the cost of an EResidual block, which consist of two group convolutions and one 1×1 convolution, is as given in Equation 11.

$$2 \times \left(K^2 \cdot C_{in} \cdot \frac{C_{out}}{G} \cdot F^2\right) + C_{in} \cdot C_{out} \cdot F^2 \tag{11}$$

Hence, by changing a residual block to our efficient block, we can reduce the computation by the ratio of

$$\frac{2 \times \left(K^2 \cdot C_{in} \cdot \frac{C_{out}}{G} \cdot F^2\right) + C_{in} \cdot C_{out} \cdot F^2}{2 \times \left(K^2 \cdot C_{in} \cdot C_{out} \cdot F^2\right)} = \frac{1}{G} + \frac{1}{2K^2}. \tag{12}$$

Because we use a kernel size as 3×3 for all convolutional layers, and the number of the channels is constant (*i.e.* 64) except the entry, exit, and upsampling block, the EResidual block reduces 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 perform an extensive case study in Section 4.3. To further reduce the parameters, we apply a technique similar to the one used by the recursive network. In other words, we force the EResidual block to be shared in the cascading block, so only one-third of the parameters are needed compared to the standard block. Fig. 4 (b) shows our efficient cascading block after applying the recursive scheme. The solid color boxes illustrate the standard modules and the hatched ones show shared components.

3.4. Comparison to Recent Models

3.4.1. Comparison to MemNet

Although MemNet [18] and ours have similar motivation, there are two main distinctions from our schemes. **1)** Feature fusion is done in a different location and manner. For instance, MemNet fuses the output features of each recursive unit at the end of the memory blocks. On the other hand, we gather the information at every possible site in the local block, thus can boost up the representation power via additional layers. **2)** MemNet takes an *early-upsample* approach which upsamples the image before putting it on the model. Although it becomes easier to implement residual learning, it worsens the model efficiency substantially. In contrast, our model gets LR images and intermediate features are upsampled at the end of the network, which enables us to accomplish a good balance between performance and efficiency.

3.4.2. Comparison to DenseNet Variants

SRDenseNet [6] and RDN [7] use a densely connected block and skip connection. Although the overall design concept can be similar, our model has two main advantages. **1)** In our models, the output of each block is associated with a global cascading connection which is a generalized form of the skip connection. In SRDenseNet and RDN, all levels of features are combined after the final dense block, but our global cascading scheme connects all blocks, which behaves as a multi-level skip connection. **2)** The connectivity schemes that we use are economical for both memory and speed. In a densely connected block, output features are concatenated to the previous information and merged at the end of the block. Because of the nature of this block, SRDenseNet and RDN require a huge burden of computation cost. In contrast, we incorporate features using an additional 1×1 convolution layer at each concatenation point, which facilitates composing more lightweight models.

3.5. Implementation Details

For the PCARN generator, we set both B and U as three, and the number of channels in all convolutional layers, except for the first, last layer and upsample block, to 64. For the upsampling unit, we use the *PixelShuffle* layer following the convolutional layer that are used in ESPCN [14]. Our discriminator network has nine convolutional layers as depicted in Fig. 1. We train our models with ADAM 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 at 10^{-4} and is halved every 4×10^5 steps.

Following the training strategy of SRGAN [8], we first train the generator with a pixel-based loss, then fine-tune the pretrained network for 2×10^5 steps with the same settings but using GAN loss. When training the generator in a pixel-wise manner, we use the L1 loss as our loss function instead of the L2. The L2 loss is widely used in the image restoration task because of its relationship to the PSNR, but L1 provided better convergence and performance in our experiments.

We use the DIV2K dataset [39], which consists of 800 training, 100 validation, and 100 test 2K resolution images. Because of the richness of this dataset, recent SR models [4, 7] use DIV2K as well. To prepare the training input, we randomly crop images to the 48×48 LR patches and augment to horizontal flip or rotation. For the test and benchmark, Set5 [40], Set14 [41], B100 [42] and Urban100 [43] datasets are used. To enable the multi-scale training, we construct the training batch using a scale of either 2 or 4, since our model can process only a single scale for each batch. The code is publicly available online on `https://github.com/nmhkahn/PCARN-pytorch`.

4. Experimental Results and Discussions

In this section, we first present the analysis results of our model. Then, we will show the quantitative evaluation and visual comparison. For all the model analysis parts (Sec. 4.1∼4.3), we conduct experiments with a 32×32 LR patch size, running 4×10^5 steps. Furthermore, we train and evaluate each of the methods ten times and gather the mean and standard deviation to inspect the performance more accurately. One thing to note here is that we represent the number of operations by Mult-Adds. It is the number of composite multiply-accumulate operations for a single image. We assume the HR image to be 720p (1280×720) to calculate Mult-Adds.

Ground-truth HR PCARN (L1) PCARN

Figure 5: **Visual comparison of adversarial training.** We compare the results trained with PCARN and without the adversarial training, PCARN (L1).

To evaluate the performance on the quantitative view, we use LPIPS and NIMA scores. Both metrics are designed to capture the perceptual quality. We use the LPIPS settings of the AlexNet with the fine-tuned linear layer (AlexNet-linear, version 0.1), and MobileNet for the NIMA backbone model.

4.1. Model Design Analysis

To investigate the performance of the proposed methods, we analyze our models via ablation study. We select the baseline to be the SRResNet. Other models have the same topology except for the inherent modules (such as the additional 1×1 convolution) that are needed for each particular architecture. Thus, the overall number of parameters is increased by up to 15% from the baseline. Table 1 shows the model analysis on the effect of cascading modules and the global residual learning paradigm.

Table 1: Analysis of the Effect of Model Design Choices. *Local*, *Global* and *L/G* means the models with local, global, and both cascading connection respectively.

Model	**Params**	**PSNR**	**SSIM**
Baseline	963K	28.42±0.02	0.7773±3e-4
+ Local	1,074K	28.45±0.01	0.7780±3e-4
+ Global	1,000K	28.47±0.02	0.7787±4e-4
+ L/G	1,111K	28.49±0.02	0.7788±3e-4
+ Residual	1,111K	**28.50±0.01**	**0.7792±3e-4**

We see that the model with local cascading works better than the baseline since this scheme carries mid- to high-level frequency signals inside the block more fluently.

Moreover, this approach conveys not only the inputs but also the mixture of intermediate features to the next block, which leverages the multi-level representations. By incorporating multi-level representations, the model can consider a variety of information from many different receptive fields when reconstructing the image. We observed higher performance gain with the global cascading scheme. This is because the advantages of the local scheme are limited to each block, which lessens the model's ability to exploit the cascading effect. One major benefit of the global cascading is that it allows for information integration from lower layers, and this information shortcut provides useful clues needed to reconstruct the HR image in the upsampling and final reconstructing processes.

In addition, we employ the global residual learning scheme shown in many recent SR methods [2, 4]. The benefit of using it can be minor, since the roles of the global cascading and residual learning overlap. However, we choose to use since it does improve performance with negligible overhead to the operation size.

Table 2: Model analysis study on perceptual perspective. MSD indicates PCARN with multi-scale discriminator.

Model	LPIPS	PSNR	SSIM
PCARN (L1)	0.2888±1e-3	28.50±0.01	0.7792±3e-4
+ GAN	0.1615±5e-3	26.36±0.25	0.7120±6e-3
+ MSD	**0.1547±2e-3**	26.10±0.17	0.6980±8e-3

To build a photo-realistic SR model, we must design a well-functioning discriminator. To see how the choice of discriminator affects the overall performance, we conducted a series of comparisons across the various types of discriminator loss. The LPIPS is used to measure the perception quality.

As shown in Table 2, PCARN with adversarial training (*+GAN*) outperforms the baseline by a large margin. Fig. 5 also exhibits the advantage of using GAN, where it successfully recovers the fine details and generates more photo-realistic images. However, since the magnitude of the pixel-level signal is diminished, the performance of pixel-based metrics (PSNR and SSIM) is degraded. Moreover, the overall training process is substantially unstable and shows high variance in all metrics. Using the multi-scale discriminator (*+MSD*) also gives an additional gain to the LPIPS. This is mainly because the signals of multiple discriminators are well-fused across different scales. Thus, it allows the model to generate detail-preserving output while maintaining the overall structure.

4.2. Initialization Strategy

Initializing the network in an appropriate manner is the key component for boosting the performance. To verify what is best for our models, we performed an experiment comparing the benchmark to six common initialization schemes: uniform, and normal distribution with various settings, as shown in Table 3. Note that we conduct this experiment using the PCARN with L1 loss, since when training a model with GAN loss, we use the pre-trained network (with L1 loss) as the starting point. Interestingly, the MSRA initialization [44], given by the equation $1.0 \times \mathrm{N}(0, \sqrt{2/F})$, and the high-range uniform initialization performed poorly against others. We argue that narrow 1×1 convolution affects the quality of initialization since the high-variance initial values tend to result in high-variance performance. On the other hand, multiplying the initial random values by 0.1 degrades the performance, since it makes the model converge too early.

Table 3: Effect of the initialization. F denotes the number of the input channels (fan-in). U/N are uniform/normal distribution.

Initialization	PSNR	SSIM
$0.1 \times \mathrm{N}(0, \sqrt{2/F})$	28.46±0.01	0.7781±3e-4
$1.0 \times \mathrm{N}(0, \sqrt{2/F})$	28.45±0.02	0.7777±3e-4
$0.1 \times \mathrm{U}(\pm\sqrt{6/F})$	28.47±0.01	0.7782±3e-4
$1.0 \times \mathrm{U}(\pm\sqrt{6/F})$	28.44±0.01	0.7778±2e-4
$0.1 \times \mathrm{U}(\pm\sqrt{1/F})$	28.45±0.02	0.7775±4e-4
$1.0 \times \mathrm{U}(\pm\sqrt{1/F})$	**28.50±0.01**	**0.7791±3e-4**

Fig. 6 supports the hypothesis as well. We plot the smoothed histogram of the binned initialized values for various initialization schemes. The gap between the ResNet with MSRA initialization (black solid) and ours (blue dots) is significant. The main reason is the difference in the channel size between the ResNet and ours: our models have considerably fewer channels for all the convolutional layers, which leads to the high standard deviation. In addition, the use of 1×1 convolution intensifies the issue since it has only one-ninth of the input channels compared to the 3×3 layers. The shape of the initialization scheme with $1.0 \times \mathrm{U}(\pm\sqrt{1/F})$ (green dash) best matches that of the ResNet.

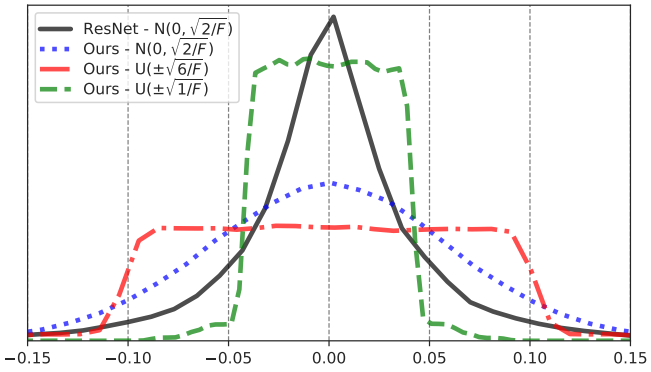


Figure 6: **Distribution of the randomly initialized networks.** We plot 100K parameters for each model using 1000 bins. Each of the parameter distributions is from the diverse sources because of the variation in the number of input channels.

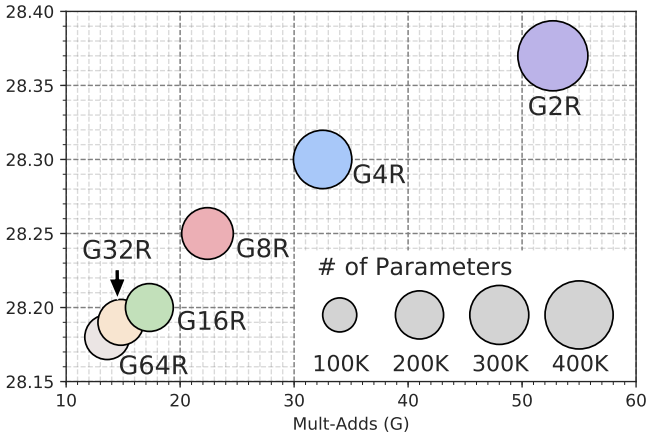


Figure 7: **Efficiency analysis of efficient models.** We evaluate all models on Set14 with $\times 4$ scale. G represents the group size of the group convolution, and R means the model with the recursive network scheme.

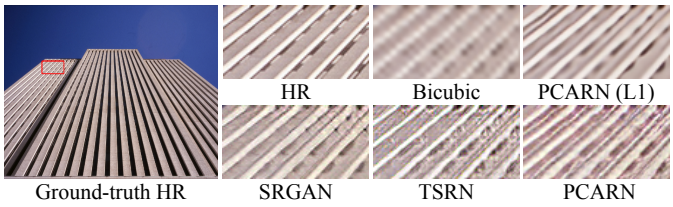


Figure 8: **A failure case.** Our perception-based model cannot is not able to reconstruct the details without sufficient information when constructing a dense structure.

4.3. Efficiency Trade-off

Fig. 7 depicts the trade-off analysis between the performance and efficiency of the efficient PCARN that uses convolution and a recursive scheme. Similarly to Sec. 4.2, we use the model with L1 loss as well, and evaluate using pixel-based metrics such as PSNR and SSIM. Although all efficient models perform worse than the PCARN, the number of parameters and operations are decreased dramatically. We choose *G4R* as the best-balanced model, which we denote as PCARN-M (mobile), since the effect of compressing the model is reduced when group size is larger than four. As a result, PCARN-M reduces the number of parameters by four times and the number of operations by nearly three times with a 0.20 loss in PSNR and 0.0053 in SSIM, compared to the PCARN.

In addition to our trade-off analysis, we also observed that depthwise separable convolution extremely degrades the performance. There can be many explanations why such an observation occurs, but we suspect that this is because the image recognition and generation tasks are entirely different, so applying group and depthwise convolution, which are mainly used in recognition fields, to the image generation domain is not fully discovered yet. Therefore, creating efficient SR model (or image generation model in general) needs more investigation with plenty of room to improve performance.

4.4. Quantitative Comparisons

We compare the proposed methods with both pixel-based and perception-based methods including SRCNN [1], MSLapSRN [19], CARN [13], SRResNet [8], SRGAN [8], EnhanceNet [20] (shortly ENet) and TSRN [21] (we choose TSRN-G since it performs better in LPIPS and NIMA). We conduct a benchmark test on the Set14 [41], B100 [42] and Urban100 [43] datasets using LPIPS [27] and NIMA [26] metrics.

Tables 4 and 5 depict the quantitative comparisons for the $\times 4$ scale datasets using LPIPS and NIMA respectively. Among all the methods, our PCARN and PCARN-M have the least Mult-Adds and a similar number of parameters with the other algorithms. With the limited resources, our models outperform all the competitors in LPIPS metric and show comparable results on the NIMA score. In detail, all pixel-based models underperform perceptually even though the pixel-based metrics show good performance. For those who fall into the perception-based category, SRGAN, ENet, and TSRN-G have analogous number of the parameters and MultAdds since they design the generator similar to SRResNet. On the other hand, by both enhancing the generator and discriminator, our method shows the best performance on the LPIPS metric with a lightweight network in terms of the number of the operations. Furthermore, PCARN-M shows comparable results only using one-fourth of both parameters and operations. For the NIMA metric, our PCARN shows comparable results with the TSRN-G, and PCARN-M achieves akin performance to the other competitors.

We also report the perception score on the various scale factors in Table 6. Note that our models are capable of processing multiple scale factors with a single network, which requires more parameters than the other methods that have similar complexity. However, without the multi-scale approach, each scale-specific networks have to be stored, so that the overall parameter is increased eventually. To the best of our knowledge, this characteristic is the first attempt at the perception-based SR task.

4.5. Visual Comparison with State-of-the-arts

In Fig.10, we illustrate the qualitative comparisons of our methods for the various $\times 4$ scale datasets. It can be seen that our models work better than others and accurately reconstructs not only the linear patterns but produce more photo-realistic textures, such as the mane of the lion and pebbles on the ground. Moreover, the proposed networks also generate cleaner outputs while other perception-based methods suffer visually artifacts.

To investigate how our models can generate SR image from different domains, we examine the visual comparison on text images using the manga109 [45] dataset. Since humans can easily distinguish high-frequency details, it is important to adequately recover the edge region in this task .Here, our method effectively restores various texts, even those with very small fonts that barely recognizable on the bicubic results.

Table 4: Comparison for ×4 SISR on LPIPS metric (AlexNet-linear version 0.1). Lower score is better. (Above) Pixel-based SR methods. (Below) Perception-based SR methods including ours.

Model	Params	MultAdds	Set5	Set14	B100	Urban
SRCNN [1]	57K	52.7G	0.2010	0.3145	0.4103	0.3161
MSLapSRN [19]	222K	435.9G	0.1777	0.2988	0.3893	0.2522
CARN [13]	1,589K	90.9G	0.1767	0.2903	0.3814	0.2366
SRResNet [8]	1,543K	127.8G	0.1729	0.2839	0.3753	0.2259
SRGAN [8]	1,543K	127.8G	0.0881	0.1741	0.2028	0.1558
ENet [20]	1,073K	120.6G	0.0994	0.1620	0.2102	0.1708
TSRN-G [21]	1,073K	120.6G	0.0881	0.1549	0.1962	0.1537
PCARN	1,589K	90.9G	**0.0745**	**0.1449**	**0.1875**	**0.1521**
PCARN-M	412K	32.5G	0.0796	0.1502	0.1981	0.1671

Table 5: Comparison for ×4 SISR on NIMA metric (MobileNet). Higher score is better. (Above) Pixel-based SR methods. (Below) Perception-based SR methods including ours.

Model	Params	MultAdds	Set5	Set14	B100	Urban
SRCNN [1]	57K	52.7G	4.316±1.629	4.372±1.694	4.340±1.695	4.624±1.642
MSLapSRN [19]	222K	435.9G	4.669±1.583	4.797±1.660	4.609±1.659	5.039±1.602
CARN [13]	1,589K	90.9G	4.731±1.564	4.908±1.639	4.694±1.640	5.149±1.599
SRResNet [8]	1,543K	127.8G	4.743±1.574	4.910±1.643	4.685±1.643	5.135±1.599
SRGAN [8]	1,543K	127.8G	4.865±1.560	5.002±1.616	4.936±1.634	5.191±1.590
ENet [20]	1,073K	120.6G	4.859±1.584	5.127±1.635	5.067±1.649	5.201±1.594
TSRN-G [21]	1,073K	120.6G	**5.052±1.569**	**5.219±1.609**	**5.168±1.613**	**5.273±1.581**
PCARN	1,589K	90.9G	4.929±1.534	5.063±1.603	5.077±1.611	5.243±1.581
PCARN-M	412K	32.5G	4.865±1.533	4.981±1.611	5.010±1.616	5.180±1.578

Table 6: Records of our models in diverse scales on LPIPS and NIMA metrics.

Scale	Model	Set5	Set14	B100	Urban
		LPIPS / NIMA	LPIPS / NIMA	LPIPS / NIMA	LPIPS / NIMA
2	PCARN (L1)	0.0546/4.669±1.592	0.0939/4.873±1.645	0.1455/4.719±1.653	0.0665/5.121±1.597
	PCARN-M (L1)	0.0549/4.647±1.595	0.0961/4.844±1.649	0.1459/4.712±1.653	0.0748/5.089±1.598
	PCARN	0.0187/4.872±1.547	0.0453/5.118±1.615	0.0598/4.980±1.634	0.0400/5.240±1.586
	PCARN-M	0.0227/4.845±1.575	0.0487/5.044±1.628	0.0643/4.936±1.641	0.0468/5.204±1.585
3	PCARN (L1)	0.1251/4.685±1.585	0.2097/4.828±1.636	0.2863/4.646±1.645	0.1612/5.108±1.592
	PCARN-M (L1)	0.1278/4.656±1.585	0.2144/4.773±1.639	0.2908/4.619±1.647	0.1769/5.060±1.592
	PCARN	0.0437/4.867±1.542	0.0962/5.122±1.601	0.1294/5.042±1.623	0.0951/5.210±1.576
	PCARN-M	0.0528/4.787±1.557	0.1061/5.003±1.604	0.1386/4.948±1.629	0.1076/5.170±1.575
4	PCARN (L1)	0.1769/4.750±1.567	0.2879/4.907±1.639	0.3780/4.704±1.639	0.2350/5.146±1.599
	PCARN-M (L1)	0.1793/4.708±1.573	0.3846/4.856±1.645	0.2928/4.656±1.643	0.2528/5.092±1.598
	PCARN	0.0745/4.929±1.534	0.1449/5.063±1.603	0.1875/5.077±1.611	0.1521/5.243±1.581
	PCARN-M	0.0796/4.865±1.533	0.1502/4.981±1.611	0.1981/5.010±1.616	0.1671/5.180±1.578

Figure 9: **Visual comparison on text image.** We compare perception-based methods that have similar complexity to ours on comic book dataset, manga109 [45] (×4 scale).

Our GAN-based PCARN can produce sharp and realistic images. However, in some dense structures like Fig. 8, it generates undesirable artifacts, unlike the L1 loss-based PCARN. We suspect that this is a common limitation shared by most, if not all, GAN-based algorithms as well because of the perception-distortion trade-off [25].

4.6. Execution time

While MultAdds can reflect the heaviness of the model well, there still exists a misalignment between the true execution times, especially when the models are run on GPUs. To investigate how our models are efficient in the real devices, we evaluate the runtime on the same machine with 3.3 GHz Intel i5 CPU (32GB RAM) and NVIDIA TITAN X GPU (11GB Memory). We use the input of the network as a 720p HR size. Fig. 11 shows the execution time of the recently proposed models benchmarked on CPU and GPU using NIMA and LPIPS metrics. In this experiment, we examine computationally-heavy model such as ESRGAN [11], G-MGBP [46] and EPSR [12] as well.

For CPU execution (above row in Fig. 11), the speed of our PCARN is faster than the other algorithms such as SRGAN and SRResNet, but produce a better result and comparable with the EPSR and G-MGBP. Our PCARN-M network is the fastest, while on a par with the heavy models. Such illustration is also reflected by the NIMA metric. The PCARN and PCARN-M methods can get good results at a relatively low computational cost.

However, unlike to the CPU results, our methods don't show such improvement on the GPU runtime (below row in Fig. 11). In fact, our models show slightly worse execution time than the ENet and TSRN-G. The reason is mainly due to the distinct characteristic of CPU and GPU environments. For example, as empirically proved in Ma et al. [35], memory fragmentation reduces the parallelism which worsens the GPU speed a lot. In our case, the cascading mechanism hinders GPU parallelism so that both PCARN and PCARN-M has fewer advantages on the GPU setting. Furthermore, group convolution used in PCARN-M is related to the GPU speed as well, diminish the speed gap between PCARN and PCARN-M. In future work, we would like to improve our framework to GPU-friendly network by carefully modifying such modules and convolution.

5. Conclusion

In this work, we proposed a deep convolutional network with a cascading scheme for fast and accurate image super-resolution. The main idea is adding multiple cascading connections starting from each intermediary layer to the others in local and global levels. In addition, we enhance our model by using a multi-scale discriminator and achieved improved performance over the recent models that have complexity analogous to ours. Also, the produced SR images are perceptually more convincing to human eyes. All the experiments were conducted on the

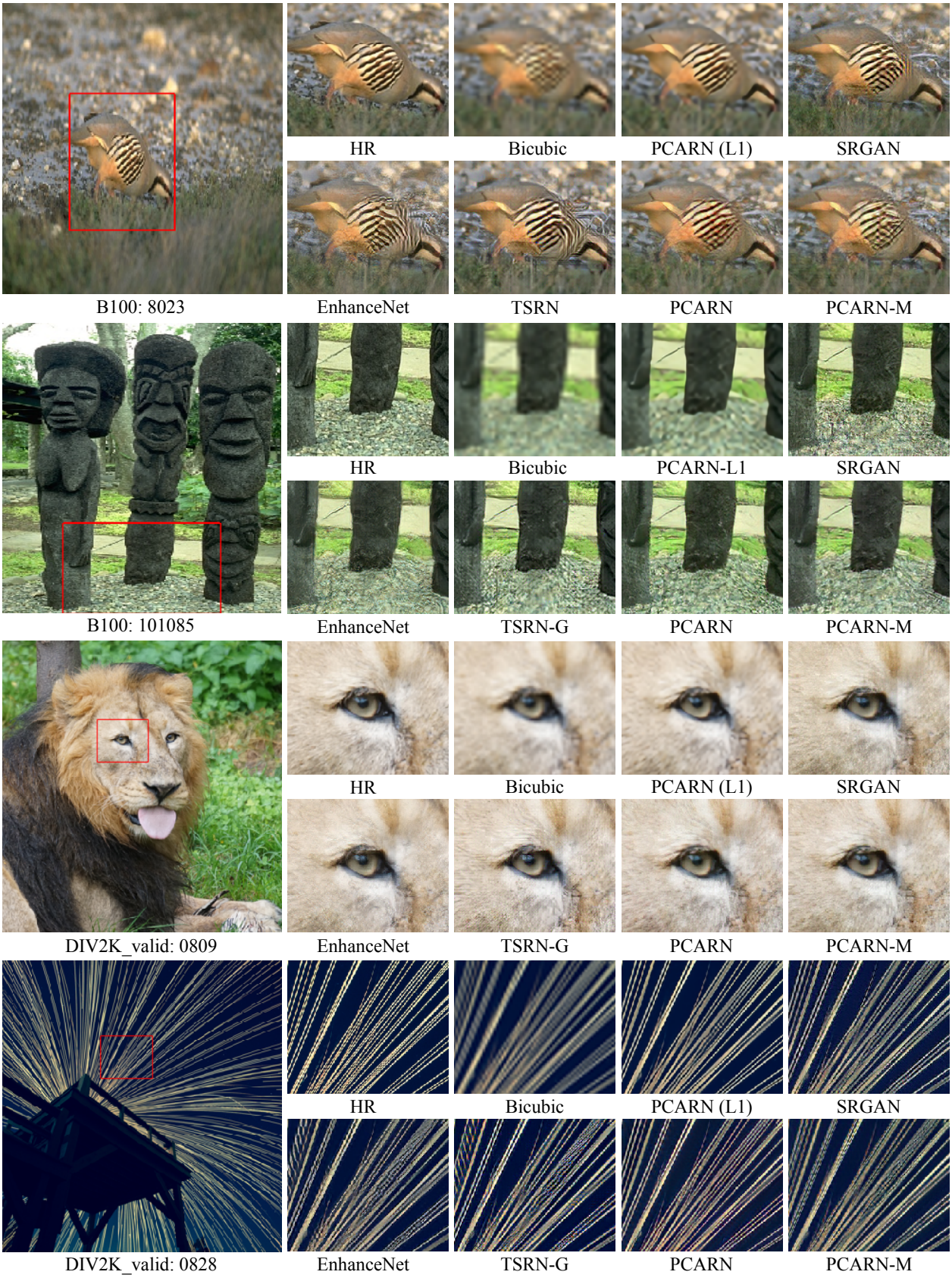


Figure 10: **Visual comparison.** We compare perception-based methods that have similar complexity to ours on the ×4 scale SR datasets.

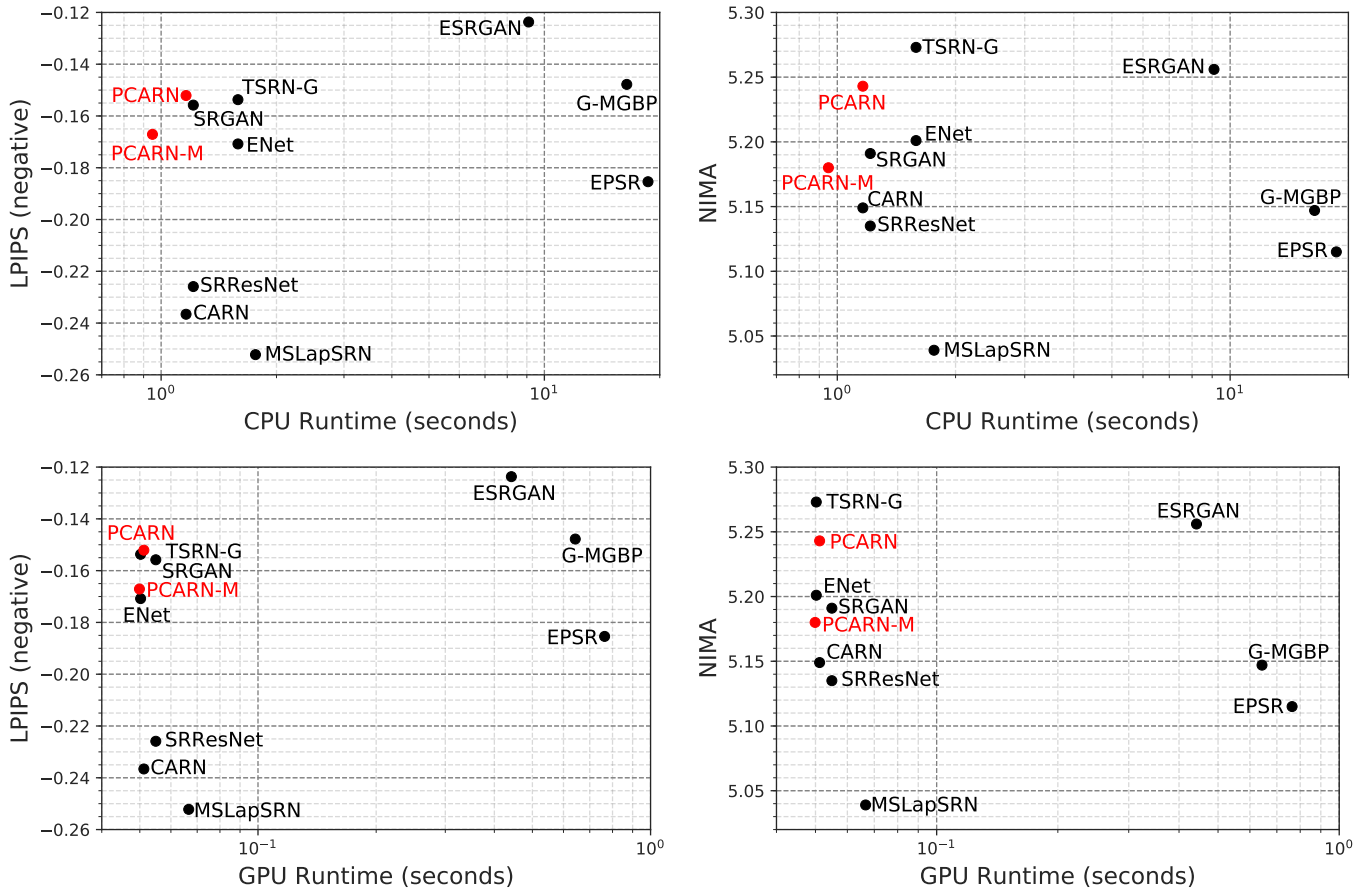


Figure 11: **Execution time on the CPU/GPU environments.** We measure the the runtime on CPU and GPU settings with LPIPS and NIMA score. We use negative LPIPS to match the direction of y-axis to the NIMA.

super-resolution field, but we hope that our work can potentially be applied to other image restoration subjects as well.

References

[1] C. Dong, C. C. Loy, K. He, X. Tang, Learning a deep convolutional network for image super-resolution, in: European conference on computer vision, Springer, 184–199, 2014.

[2] 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, 1646–1654, 2016.

[3] 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, 624–632, 2017.

[4] B. Lim, S. Son, H. Kim, S. Nah, K. Mu Lee, Enhanced deep residual networks for single image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 136–144, 2017.

[5] 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, 1637–1645, 2016.

[6] 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, 4799–4807, 2017.

[7] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, Y. Fu, Residual dense network for image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2472–2481, 2018.

[8] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network., in: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2672–2680, 2014.

[10] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: European Conference on Computer Vision, Springer, 694–711, 2016.

[11] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, X. Tang, Esrgan: Enhanced super-resolution generative adversarial networks, arXiv preprint arXiv:1809.00219 .

[12] S. Vasu, N. Thekke Madam, A. Rajagopalan, Analyzing perception-distortion tradeoff using enhanced perceptual super-resolution network, in: Proceedings of the European Conference on Computer Vision (ECCV), 0–0, 2018.

[13] N. Ahn, B. Kang, K.-A. Sohn, Fast, accurate, and lightweight super-resolution with cascading residual network, in: Proceedings of the European Conference on Computer Vision (ECCV), 252–268, 2018.

[14] 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, 1874–1883, 2016.

[15] C. Dong, C. C. Loy, X. Tang, Accelerating the super-resolution convolutional neural network, in: European conference on computer vision, Springer, 391–407, 2016.

[16] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, C. Schroers, A Fully Progressive Approach to Single-Image Super-Resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 864–873, 2018.

[17] N. Ahn, B. Kang, K.-A. Sohn, Image super-resolution via progressive cascading residual network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 791–799, 2018.

[18] Y. Tai, J. Yang, X. Liu, C. Xu, Memnet: A persistent memory network for image restoration, in: Proceedings of the IEEE international conference on computer vision, 4539–4547, 2017.

[19] W.-S. Lai, J.-B. Huang, N. Ahuja, M.-H. Yang, Fast and accurate image super-resolution with deep laplacian pyramid networks, IEEE transactions on pattern analysis and machine intelligence .

[20] M. S. Sajjadi, B. Schölkopf, M. Hirsch, Enhancenet: Single image super-resolution through automated texture synthesis, in: Proceedings of the International Conference on Computer Vision (ICCV), 2017.

[21] M. W. Gondal, B. Schölkopf, M. Hirsch, The unreasonable effectiveness of texture transfer for single image super-resolution, arXiv preprint arXiv:1808.00043 .

[22] L. A. Gatys, A. S. Ecker, M. Bethge, Image style transfer using convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2414–2423, 2016.

[23] A. Jolicoeur-Martineau, The relativistic discriminator: a key element missing from standard GAN, arXiv preprint arXiv:1807.00734 .

[24] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al., Image quality assessment: from error visibility to structural similarity, IEEE transactions on image processing 13 (4) (2004) 600–612.

[25] Y. Blau, T. Michaeli, The perception-distortion tradeoff, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6228–6237, 2018.

[26] H. Talebi, P. Milanfar, Nima: Neural image assessment, IEEE Transactions on Image Processing 27 (8) (2018) 3998–4011.

[27] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The Unreasonable Effectiveness of Deep Features as a Perceptual Metric, in: CVPR, 2018.

[28] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 1097–1105, 2012.

[29] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv preprint arXiv:1510.00149 .

[30] 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 preprint arXiv:1704.04861 .

[31] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531 .

[32] 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, arXiv preprint arXiv:1602.07360 .

[33] L. Sifre, S. Mallat, Rigid-motion scattering for image classification, Ph.D. thesis, Citeseer, 2014.

[34] X. Zhang, X. Zhou, M. Lin, J. Sun, ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, in: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[35] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[36] 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, 770–778, 2016.

[37] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 .

[38] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, B. Catanzaro, High-resolution image synthesis and semantic manipulation with conditional gans, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 8798–8807, 2018.

[39] E. Agustsson, R. Timofte, Ntire 2017 challenge on single image super-resolution: Dataset and study, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 126–135, 2017.

[40] M. Bevilacqua, A. Roumy, C. Guillemot, M. L. Alberi-Morel, Low-complexity single-image super-resolution based on nonnegative neighbor embedding .

[41] J. Yang, J. Wright, T. S. Huang, Y. Ma, Image super-resolution via sparse representation, IEEE transactions on image processing 19 (11) (2010) 2861–2873.

[42] D. Martin, C. Fowlkes, D. Tal, J. Malik, et al., A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, Iccv Vancouver:, 2001.

[43] 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, 5197–5206, 2015.

[44] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 1026–1034, 2015.

[45] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, K. Aizawa, Sketch-based manga retrieval using manga109 dataset, Multimedia Tools and Applications 76 (20) (2017) 21811–21838.

[46] P. Navarrete Michelini, H. Liu, D. Zhu, Multi–Scale Recursive and Perception–Distortion Controllable Image Super–Resolution, in: The European Conference on Computer Vision Workshops (ECCVW), 2018.