This CVPR workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

# Image Super-resolution via Progressive Cascading Residual Network

Namhyuk Ahn Byungkon Kang Kyung-Ah Sohn
Department of Computer Engineering, Ajou University
{aa0dfg,byungkon,kasohn}@ajou.ac.kr

## Abstract

*The problem of enhancing the resolution of a single low-resolution image has been popularly addressed by recent deep learning techniques. However, many deep learning approaches still fail to deal with extreme super-resolution scenarios because of the instability of training. In this paper, we address this issue by adapting a progressive learning scheme to the deep convolutional neural network. In detail, the overall training proceeds in multiple stages so that the model gradually increases the output image resolution. In our experiments, we show that this property yields a large performance gain compared to the non-progressive learning methods.*

## 1. Introduction

Given a low-resolution image, the task of reconstructing it to a high-resolution image is known as super-resolution (SR). In this research, we focus on the single image super-resolution (SISR) task, where the SR operation is performed on a single given low-resolution image. Since the conversion of a low-resolution image to a higher-resolution one is a one-to-many mapping, it is very difficult to achieve high-quality super-resolution from a single image. However, SISR can provide a variety of useful opportunities such as in the field of real-time video streaming or surveillance system, and hence it is a very active research area.

With the advancement of deep learning technologies in various computer vision tasks [10, 22, 20], many deep learning-based models have been proposed to tackle the SISR task [7, 13, 26, 19]. The sizes of such models vary greatly, ranging from as few as three convolutional layers (SRCNN: [7]) to more than hundred layers (MDSR: [19]), and the overall performance has grown as well. However, since most of the methods reconstruct an HR image by a single upsampling step, such deep architectures can degrade the performance in an extreme SR cases such as $\times 8$ scale task. Most other SR researches only focus on smaller scale resolutions, such as $\times 2, 3, 4$. There are not many studies that deal with extreme scales of more than $\times 8$, since it is too hard to infer the extra information needed to upscale an image to such a high resolution. The issue of degradation in extreme SR tasks can be more critical if there are not enough refinement layers after the upsampling process. Recent methods [23, 19, 2] also can suffer from this problem since they upsample the input image at the end of the network.


Figure 1: **Super-resolution result of our proposed progressive CARN model compared to the existing algorithms.** (NTIRE2018 challenge [28] task1: $\times 8$ scale)

The difficulty of training for the extreme SR cases comes from the issue of instability in training. This phenomenon occurs when the upsampling is performed at the end of the network, which is a common network design choice that

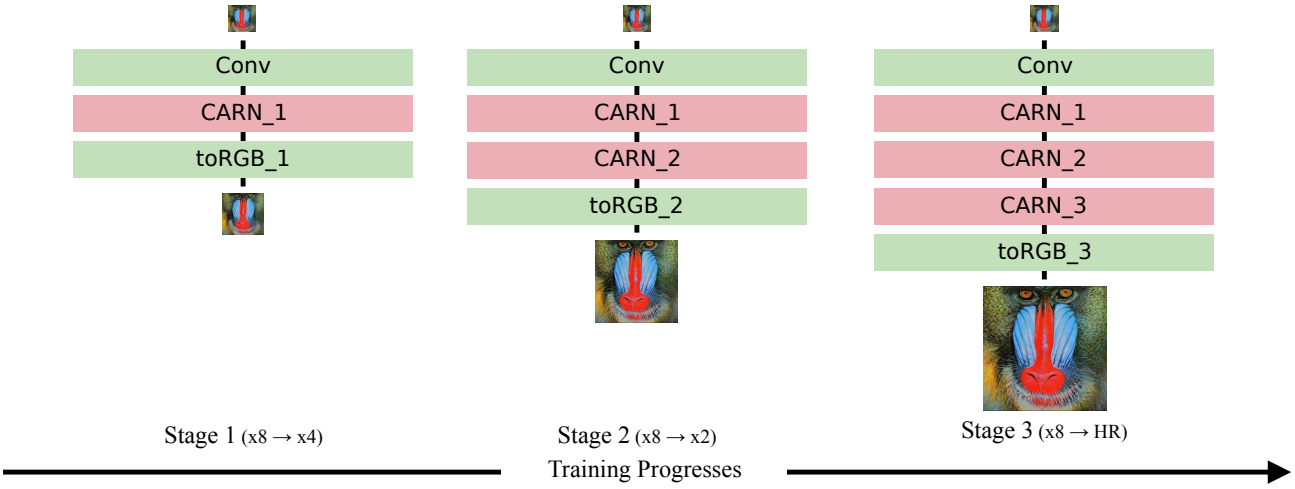


Figure 2: **Illustration of progressive training for the $\times 8$ scale super-resolution task**. The number that is denoted in each layer name means the stage number in which the corresponding module is added. For instance, CARN_2 denotes the CARN module that is added in stage two.

most of the recent methods adopt. This gives rise to the sudden shock to the model when image comes to the refinement block so that it causes the instability of training. Furthermore, the overall quality of SR image can be degraded since not enough refining process is applied after the upsampling. Especially, the quality issue is more striking when performing the extreme SR cases.

To alleviate the difficulty of training a deep convolutional network, one can train a model by a layer-wise training approach. For example, VGGNet [24] first trains a smaller convolutional network and gradually increases the depth of the network to avoid instability of the training. Another approach is adding the auxiliary classifier at the middle of the network to help the information flow of the earlier layer [25]. However, training a very deep network is still an unsolved and a challenging problem.

In this paper, we propose a progressive cascading residual network which applies the progressive learning [12] approach to the cascading residual network [2]. Our primary scheme is a training method for extreme SR cases, where we generate a relatively low-resolution output at first, and then progressively increase the output resolution by adding an extra network to our model. This scheme alleviates the instability of training since it can reduce the sudden size change of the model by gradually upsampling the image at the middle of the network. Moreover, the quality degradation issue is reduced by the same mechanism that the instability issue is solved. The instability issue caused by training deep convolutional network can also be solved by our proposed model. This is because the progressive learning works as a fine-tuning of the pre-trained network.

In summary, we propose a progressive cascading residual network (progressive CARN) method for the extreme super-resolution. It applies the progressive learning to the cascading residual network to make the overall training procedure more stable. By doing so, this model mitigates the instability of training caused by the training of very deep convolutional neural network on extremely low-resolution inputs. Furthermore, our work also helps to avoid the quality degradation problem by performing the upsampling process periodically.

## 2. Related Work

### 2.1. Deep Learning Based Image Super-Resolution

After the success of AlexNet [16] in image recognition task [5], various deep learning approaches have been applied to many computer vision tasks including SR [7, 13, 26]. The first deep learning-based method, SRCNN [7] outperformed traditional SR algorithms. However, SRCNN has settled for shallow layers because of the difficulty in training. To better harness the depth of deep learning models, Kim et al. [13] proposed VDSR, which uses *residual learning* to map 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.*, $\mathbf{y} = \mathbf{x} + \mathbf{r}$.

The aforementioned methods have a large number of operations compared to its depth, as input images are upsampled (usually bicubic interpolation) before being fed into the network. Taking a different approach from those, FSRCNN [8] and ESPCN [23] upsample images at the end of the networks. This approach leads to the reduction in the number of operations compared to the former ones. However, as pointed out by Lim et al. [19], models using

this approach cannot be applied to multi-scale training as shown in VDSR [13]. To resolve this, MDSR [19] and CARN [2] contain scale-specific upsampling modules in a single framework, allowing it to handle multi-scale images to appropriate scale-specific pathways.

### 2.2. Progressive Training

To further improve the quality of generated high-resolution image, many methods [6, 31, 12] apply a progressive generation scheme to a generative model, which is usually a generative adversarial network [9]. The LapGAN [6] uses a laplacian pyramid [4] so that the model generates the sub-band residual instead of a natural image. During the reconstruction phase, the model reconstructs the natural image by combining the generated residual image and upsampled input LR image. StackGAN [31] generates photo-realistic high-resolution images conditioned on text descriptions via *sketch-refinement*. This method uses a two-stage training procedure which sketches the coarse structure of the objects in the first stage and generates the high-resolution image from the intermediary image in the second stage. To generate very high-resolution images such as $1024 \times 1024$ resolution, Karras et al. [12] proposed a progressive training methodology for generative adversarial networks. The way progressive training works is that it starts from a low-resolution input and then gradually adds new layers to the model. This makes training the model much easier than the direct learning approach.

Yet, only a few deep learning based SR algorithms apply the progressive approach. Among them, the most successful one is LapSRN [17, 18]. Similarly to the LapGAN, LapSRN uses Laplacian pyramids when restoring images. To this end, from the LR input image, the model progressively increases the image resolution and generates a series of sub-band residual images. The output residual images and the upsampled LR input are combined to produce the final SR image in the reconstruction phase. To further decrease a model size, they share the weights of the component between the multi-level pyramids [18].

## 3. Our Methods

In this section, we describe the methodology of the proposed progressive CARN that uses progressive learning based on CARN. Any deep learning-based SISR model can be a backbone network of the progressive approach, but we select CARN because it achieves a good balance between efficiency and performance. In Section 3.1, we first overview the CARN model, before presenting our progressive CARN model in Section 3.2.

### 3.1. Cascading Residual Network

Cascading residual network [2] is designed based on a residual network, with a *cascading mechanism*. CARN model consists of $B$ cascading blocks and $1 \times 1$ convolution layers as shown in Figure 3 (a). The cascading blocks contain $U$ residual units and $1 \times 1$ convolution layers as shown in Figure 3 (b). Note that the default setting of $B$ and $U$ used in the original model is $B = 3$ and $U = 3$.

Here, we express the CARN module mathematically and extend the formulation to the progressive CARN in Section 3.2. Let $f$ be a convolution function and $\tau$ be an activation function. Then, we can define the $i$-th residual unit $R^i$, which has two convolutions followed by a residual addition as below.

$$R^i\left(I; W_R^i\right) = \tau\left(f\left(\tau\left(f\left(I; W_R^{i,1}\right)\right); W_R^{i,2}\right) + I\right) \quad (1)$$

Here, $I$ denotes the input of the residual unit, $W_R^i$ is the parameter set of the entire residual unit, and $W_R^{i,j}$ is the parameter of the $j$-th convolution layer in the $i$-th unit.

By Equation 1, the local cascading block with input feature map $I$ can be derived. We denote $B^{i,j}$ as the output of the $j$-th residual unit 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 $B_{local}^i$ that has $U$ residual units with parameters $W_B^i$ is denoted as

$$B_{local}^i\left(I; W_B^i\right) \equiv B^{i,U}, \quad (2)$$

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

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

By combining Equation 2 and global cascading mechanism, we can formulate $i$-th CARN module $M^i$ with parameter set $W_M^i$ as Equation 3. This module has $B$ number of cascading block, and final $\times 2$ upsampling block $\lfloor . \rfloor \uparrow_2$.

$$M^i\left(I; W_M^i\right) \equiv \lfloor H^B \rfloor \uparrow_2 \quad (3)$$

where,

$$\begin{aligned} H^0 &= I \\ H^b &= f\left(\left[H^0, \ldots, H^{b-1}, B_{local}^u\left(H^{b-1}; W_B^b\right)\right]\right) \\ &\qquad \text{for } b = 1, \ldots, B. \end{aligned}$$

The main difference between CARN and ResNet is the presence of local and global cascading modules as shown in Equation 2 and 3. Figure 3 (a) visually illustrates how the global cascading occurs. The outputs of intermediary layers are *cascaded* into the higher layers, and finally converge on a single $1 \times 1$ 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 3 (b). Local cascading is

almost identical to the global one except that the unit blocks are plain residual units.

Cascading on both the local and global levels has two advantages: 1) The model incorporates features across layers, which allows to learn multi-level representations. 2) Multi-level cascading connection behaves as multi-level shortcut connections that quickly propagate information from lower to higher layers. As a result, these properties make the model more powerful compared to the residual networks.

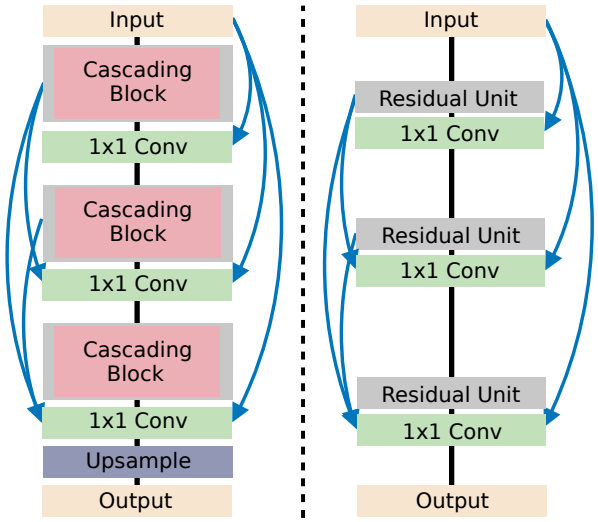


Figure 3: **Simplified structure of cascading residual network with $B = 3$ and $U = 3$.** The cascading residual network (**left**) contains cascading blocks and global cascading connections. The cascading block (**right**) consists of plain residual units and local cascading connections. In this figure, the blue arrows indicate (global or local) cascading connections.

### 3.2. Progressive Cascading Residual Network

We propose to build our model based on the CARN [2] architecture and progressive learning scheme [12] to effectively reconstruct extremely low-resolution images. The key concept of the methodology is similar to that of Karras et al. [12], but we adapt this scheme for our SR task as shown in Figure 2. In detail, we set the number of stages as three in the $\times 8$ scale SR task. That is, in each stage, the model performs $\times 8 \rightarrow \times 4$, $\times 8 \rightarrow \times 2$, and $\times 8 \rightarrow$ HR tasks sequentially.

The training starts from stage one, which produces the $\times 4$ scale image from the first CARN module and the corresponding reconstruction block named *toRGB_1*, as shown in Figure 2. The *toRGB* block is a convolution layer that refines the details of the upsampled image. After the end of the first stage, we add extra CARN modules to the model and replace the previous reconstruction block with the one that produces the image in double resolution. This training procedure iterates until it reaches the last stage. The output of the final stage is an SR image of the same size as the HR image. The overall training process is shown in Algorithm 1. Note that we use a set-like representation $S$ for the neural network we build and $\Gamma$ for the set of learning rates.

**Algorithm 1:** Overall training process

**Input:** Batch of LR and HR images $(I_{LR}, I_{HR})$,
# of stages $N$, initial learning rate $\gamma$
**Output:** Trained model $S(I_{LR}; W_S)$
$\Gamma \leftarrow \{\gamma\}$
$S \leftarrow \{f(W_c)\}$ // initial convolution layer in Fig. 2
**for** $i \leftarrow 1$ **to** $N$ **do**
  $S \leftarrow S \cup \{M^i\left(W_M^i\right)\}$ // as in Eq. 3
  $\Gamma \leftarrow \Gamma \cup \{\gamma\}$
  Attach `toRGB_i` layer to $S$ // as in Fig. 2
  $I_{SR} \leftarrow S(I_{LR}; W_S)$
  Update $W_S$ with $(I_{SR}, I_{HR})$ by corresponding $\Gamma$
  **if** $i < N$ **then**
    Detach `toRGB_i` layer from $S$
  **end**
  // decay the learning rates of previous modules
  $\Gamma \leftarrow 0.1 \times \Gamma$
**end**

To further stabilize progressive training, we reduce the learning rate of pre-trained modules ten times. This is a simpler approach than smooth network transition [12], but it also makes the training process stable. The idea behind it is the same as fine-tuning a pre-trained network. However, we found that freezing the pre-trained modules would degrade the overall quality since the information that had to be propagated to earlier blocks could not flow properly.

Applying progressive training to any SR network alleviates the *instability* of training problem. This issue comes up when SR tasks are performed using deep architectures, where the final upsampling is usually done abruptly towards the end of the network. This can cause the network's capacity unable to catch up with the sudden increase of the image being processed. In this case, the overall performance might become unstable. Our progressive training scheme alleviates this problem by introducing gradual increases in the image. We will discuss the results and model analysis of our approach in Section 4.3 and 4.4.

## 4. Experimental Results

### 4.1. Datasets

There exist diverse single image SR datasets, but the most widely used ones are the 291 image set by Yang

et al. [30] and the Berkeley Segmentation Dataset [21]. However, because these two do not have sufficient images for training a deep neural network, we use the DIV2K dataset [1, 27, 28]. 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 [19, 2] use DIV2K as well. In this paper, our goal is to produce the SR image from the LR image of the $\times 8$ scale DIV2K dataset [28]. And to construct intermediary scale images for progressive training, we use DIV2K $\{\times 2, \times 4\}$ dataset [1]. We use the standard benchmark datasets such as Set5 [3], Set14 [30], B100 [21], Urbann100 [11], and DIV2K validation set for testing and benchmarking.

### 4.2. Implementation and Training Details

We set $B = 4$ and $U = 8$ for all the CARN modules, and removed the final convolution layer after the upsampling block since the model produces RGB images at once at the end of the networks (*e.g.* at the *toRGB* module in Figure 2). For the inputs, we use RGB LR images whose patches are of size $48 \times 48$ for training. We sample the LR patches randomly and augment them with a random horizontal flip and four 90 degree rotations. We use three-stage progressive training for $\times 8$ scale SR task by setting the batch size 32, 8, and 2 for $1.5 \times 10^5$, $2.0 \times 10^5$, and $3 \times 10^5$ steps, respectively. We train our models with the ADAM optimizer [15] with $10^{-4}$ as the default learning rate. As described in Section 3.2, we decrease the learning rate of the pre-trained modules by ten times for the training stability.

To boost the performance of our model, we additionally apply geometric self-ensemble [19]. To do this, we flip and rotate LR image to make eight augmented images. Then, we generate SR images from the augmented ones and average these after recovering to its own original geometry by inverse transformation function. Despite not requiring any extra models, the performance gain is comparable to the inter-model ensemble method. From now on, we add the $+$ symbol to the name of a model to denote the self-ensemble version on the benchmark results.

### 4.3. Performance Analysis

To investigate the performance behavior of the proposed method, we analyze our model via ablation study. Table 1 presents the ablation study on the effect of progressive learning and self-ensemble methods. Here, CARN-B3U3 is the default setting of the original model and CARN-B24U4 is the enlarged version of CARN to match the number of network parameters (9.46 million) to our proposed one. This model has 24 cascading blocks and each block has four residual units. Overall, the total number of parameters is the same as that of the progressive CARN, which consists of three B8U4 CARN body modules. In addition, we also show the result of the EDSR [19] to see how progressive training can effectively handle the instability that happens when training a very deep network.

| Model | # Params. | Time | DIV2K valid |
|---|---|---|---|
| CARN-B3U3 [2] | **1.25M** | **0.14s** | 25.28 |
| EDSR [19] | 45.45M | 1.11s | 25.47 |
| CARN-B24U4 [2] | 9.46M | 0.35s | 25.36 |
| progressive CARN | 9.46M | 1.73s | 25.53 |
| progressive CARN+ | 9.46M | 13.80s | **25.64** |

Table 1: **Ablation study of the proposed progressive CARN.** We evaluate the number of parameters, running time, and the performance on DIV2K validation dataset $\times 8$ scale. The CARN is the baseline model proposed by Ahn et al. [2], and CARN-B24U4 is the enlarged version to match parameters to our progressive CARN. The progressive CARN+ is the model with geometric self-ensemble.

The CARN-B24U4 outperforms CARN-B3U3 since it has almost eight times more parameters than the latter. Our proposed progressive CARN outperforms CARN-B24U4 with a large margin with a similar number of parameters and our model achieves better performance than EDSR as well. The performance gap between previous models and ours can be attributed to the progressive training scheme. For example, CARN-B24U4 and our progressive CARN have nearly identical model sizes, yet the latter outperforms the former. So we can see that having similar model size does not guarantee similar performance.

We claim that the progressive training scheme resulted in an effect that resembles layer-wise training of a deep network. In our case, each intermediary CARN layer receives training signals in the form of LR images. By learning to enlarge the given image to an intermediary-sized images, our model is able to overcome the performance gap. In addition to the high performance, our model enjoys smaller model size compared to the EDSR model. As shown in Table 1, our model uses only $20\%$ of the parameters used by EDSR, while the performance is better.

The model with self-ensemble also shows a performance gain compared to the one without it. The running time is slower than the without-ensemble version since it cannot be run in parallel to avoid out-of-memory issue. However, it can be used in many situations since it does not require any extra models when performing ensemble. Also, the running time can be significantly reduced when we run the models in parallel.

### 4.4. Comparison with state-of-the-art Methods

We compare the proposed model with previous state-of-the-art SR methods [7, 13, 17, 18, 19, 2] on two commonly used image quality metrics: PSNR and the structural similarity index (SSIM) [29]. Table 2 shows the quantitative comparisons of the performances for $\times 8$ scale SR over the Set5, Set14, B100, and Urban100 datasets. Here, our proposed progressive CARN outperforms all methods with a large margin over all the datasets. Most of the previous algorithms [7, 13, 19, 2] tend to suffer from the instability problem during training. As mentioned above, this is because the LR images do not have sufficient information to recover, so that upsampling the LR image abruptly can fail to reconstruct the SR image. In addition, our method also shows better performance compared to the progressive upsampling approaches [17, 18]. This advantage can be achieved by progressive training which mitigates the instability of training. Furthermore, the progressive CARN+, which is the geometric self-ensemble version, achieves even better performance. These observation can also be found in the visual qualitative comparison. As shown in Figure 5, our model works better than the others and accurately reconstructs not only stripes and line patterns, but also complex objects such as alphabet type, as depicted in *ppt3* image from the Set14 dataset.

We also visually illustrate the qualitative comparisons among the CARN models and our proposed ones. Somewhat surprisingly, we often observe the degradation issues of the CARN-B24U4, especially for the complex patterns or objects. The images *0821* and *0831* from the DIV2K show the degradation problems that CARN-B24U4 experiences. However, our proposed method shows a better quality than the the others since it can be trained more stably with a progressive learning scheme.

## 5. Conclusion

In this work, we proposed a progressive cascading residual network that can perform SISR accurately even in an extreme low-resolution scenario. The main idea behind our work is to apply progressive learning scheme to cascading residual networks. By using the progressive scheme, the training process becomes much easier and more stable, since the model first learns the coarse structure and gradually learns how to restore details in the later stages. Our experiment shows that employing this idea leads to better performance on various benchmark datasets compared to the non-progressive approaches.

We wish to further improve this work by making it more efficient. One may meet this need by combining the recursive network [14, 18] with our method so that the parameters of the modules in all stages become tied.

## Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education [NRF-2016R1D1A1B03933875].

## References

[1] E. Agustsson and R. Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of CVPR Workshops*, 2017.

[2] N. Ahn, B. Kang, and K.-A. Sohn. Fast, accurate, and, lightweight super-resolution with cascading residual network. *arXiv preprint arXiv:1803.08664*, 2018.

[3] M. Bevilacqua, A. Roumy, C. Guillemot, and M. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proceedings of BMVC*, 2012.

[4] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. In *Readings in Computer Vision*, pages 671–679. Elsevier, 1987.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of CVPR*, 2009.

[6] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Proceedings of NIPS*, 2015.

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *Proceedings of ECCV*, 2014.

[8] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *Proceedings of ECCV*, 2016.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of NIPS*, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of CVPR*, 2016.

[11] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of CVPR*, 2015.

[12] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *Proceedings of ICLR*, 2018.

[13] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of CVPR*, 2016.

[14] J. Kim, J. Kwon Lee, and K. Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of CVPR*, 2016.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*, 2012.

| Dataset | Bicubic | SRCNN [7] | VDSR [13] | LapSRN [17] | MS-LapSRN [18] | CARN [2] | EDSR [19] | **progressive CARN** | **progressive CARN+** |
|---|---|---|---|---|---|---|---|---|---|
| Set5 | 24.40 / 0.658 | 25.33 / 0.690 | 25.93 / 0.724 | 26.15 / 0.738 | 26.34 / 0.753 | 26.72 / 0.766 | 27.09 / 0.781 | 27.17 / 0.782 | 27.28 / 0.786 |
| Set14 | 23.10 / 0.566 | 23.76 / 0.591 | 24.26 / 0.614 | 24.35 / 0.620 | 24.57 / 0.629 | 24.83 / 0.635 | 24.96 / 0.643 | 25.04 / 0.644 | 25.16 / 0.647 |
| B100 | 23.67 / 0.548 | 24.13 / 0.566 | 24.49 / 0.583 | 24.54 / 0.586 | 24.65 / 0.592 | 24.72 / 0.591 | 24.81 / 0.599 | 24.87 / 0.600 | 24.92 / 0.601 |
| Urban100 | 20.74 / 0.516 | 21.29 / 0.544 | 21.70 / 0.571 | 21.81 / 0.581 | 22.06 / 0.598 | 22.25 / 0.604 | 22.55 / 0.624 | 22.62 / 0.626 | 22.78 / 0.631 |

Table 2: **Quantitative results of deep learning based SR algorithms.** We evaluate PSNR / SSIM for scaling factor of $\times 8$ on the public benchmark datasets. The two rightmost columns are our methods, and progressive CARN+ denotes the geometric self-ensemble version of progressive CARN. The red text indicates the best performance and blue indicates the second best.

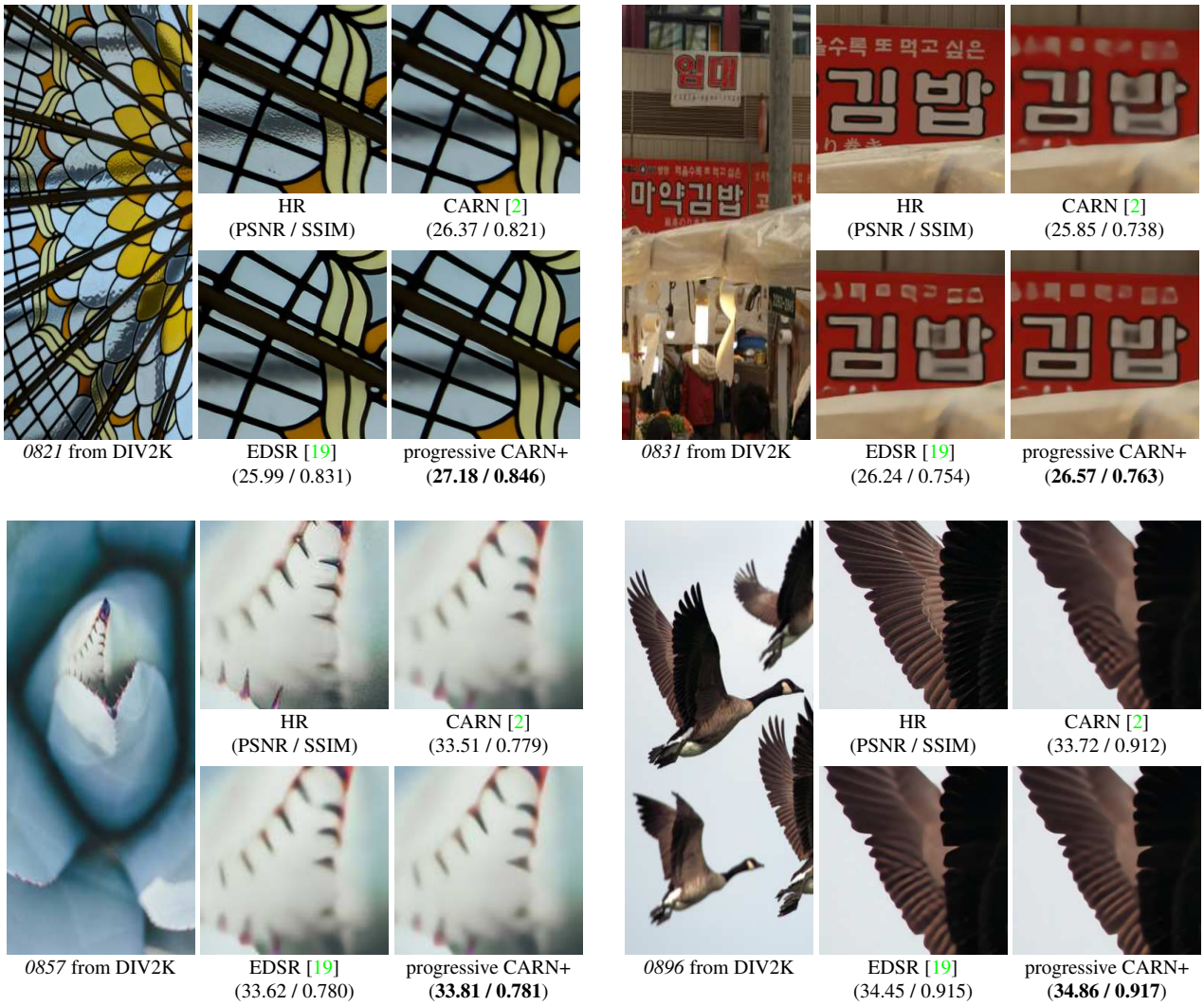


Figure 4: **Qualitative comparison on NTIRE2018 super-resolution challenge [28]**. (task1: super-resolution on $\times 8$ scale)


Figure 5: **Visual qualitative comparison for $\times 8$ SR on the Set14, B100, and Urban100 datasets.**

[17] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of CVPR*, 2017.

[18] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Fast and accurate image super-resolution with deep Laplacian pyramid networks. *arXiv preprint arXiv:1710.01992*, 2017.

[19] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of CVPR Workshops*, 2017.

[20] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of CVPR*, 2017.

[21] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of ICCV*, 2001.

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of NIPS*, 2015.

[23] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of CVPR*, 2016.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. In *Proceedings of CVPR*, 2015.

[26] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *Proceedings of CVPR*, 2017.

[27] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, et al. NTIRE 2017 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[28] R. Timofte, S. Gu, J. Wu, L. Van Gool, L. Zhang, M.-H. Yang, et al. NTIRE 2018 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[30] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[31] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of ICCV*, 2017.