Enhanced Deep Residual Networks for Single Image Super-Resolution

Bee Lim Sanghyun Son Heewon Kim Seungjun Nah Kyoung Mu Lee

Department of ECE, ASRI, Seoul National University, 08826, Seoul, Korea

forestrainee@gmail.com, thstkdgus35@snu.ac.kr, ghimhw@gmail.com

seungjun.nah@gmail.com, kyoungmu@snu.ac.kr

Abstract

*Recent research on super-resolution has progressed with the development of deep convolutional neural networks (DCNN). In particular, residual learning techniques exhibit improved performance. In this paper, we develop an enhanced deep super-resolution network (**EDSR**) with performance exceeding those of current state-of-the-art SR methods. The significant performance improvement of our model is due to optimization by removing unnecessary modules in conventional residual networks. The performance is further improved by expanding the model size while we stabilize the training procedure. We also propose a new multi-scale deep super-resolution system (**MDSR**) and training method, which can reconstruct high-resolution images of different upscaling factors in a single model. The proposed methods show superior performance over the state-of-the-art methods on benchmark datasets and prove its excellence by winning the NTIRE2017 Super-Resolution Challenge [26].*

1. Introduction

Image super-resolution (SR) problem, particularly single image super-resolution (SISR), has gained increasing research attention for decades. SISR aims to reconstruct a high-resolution image I^{SR} from a single low-resolution image I^{LR}. Generally, the relationship between I^{LR} and the original high-resolution image I^{HR} can vary depending on the situation. Many studies assume that I^{LR} is a bicubic downsampled version of I^{HR}, but other degrading factors such as blur, decimation, or noise can also be considered for practical applications.

Recently, deep neural networks [11, 12, 14] provide significantly improved performance in terms of peak signal-to-noise ratio (PSNR) in the SR problem. However, such networks exhibit limitations in terms of architecture optimality. First, the reconstruction performance of the neural network models is sensitive to minor architectural changes. Also, the same model achieves different levels of performance by different initialization and training techniques. Thus, carefully designed model architecture and sophisticated optimization methods are essential in training the neural networks.

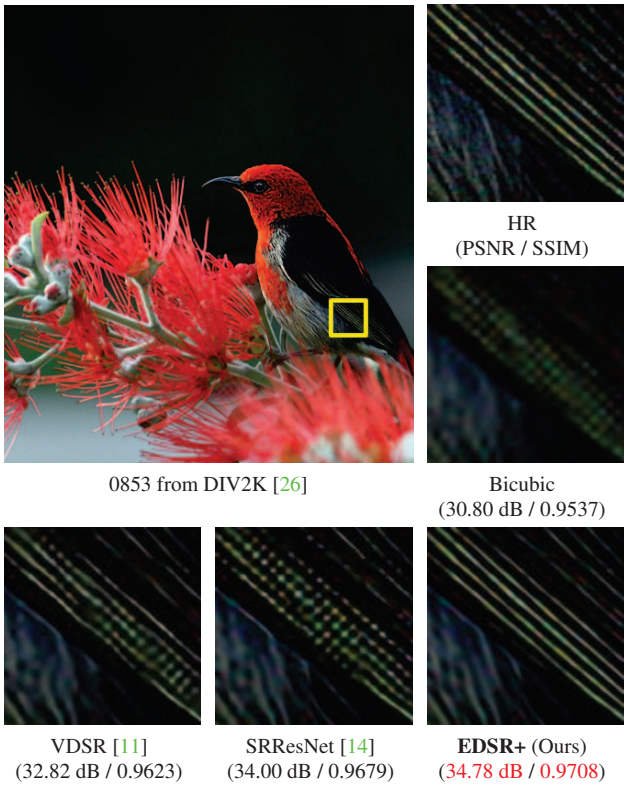


Figure 1: $\times 4$ Super-resolution result of our single-scale SR method (EDSR) compared with existing algorithms.

Second, most existing SR algorithms treat super-resolution of different scale factors as independent problems without considering and utilizing mutual relationships among different scales in SR. As such, those algorithms require many scale-specific networks that need to to be trained independently to deal with various scales. Exceptionally,

2160-7516/17 $31.00 © 2017 IEEE
DOI 10.1109/CVPRW.2017.151

VDSR [11] can handle super-resolution of several scales jointly in the single network. Training the VDSR model with multiple scales boosts the performance substantially and outperforms scale-specific training, implying the redundancy among scale-specific models. Nonetheless, VDSR style architecture requires bicubic interpolated image as the input, that leads to heavier computation time and memory compared to the architectures with scale-specific upsampling method [5, 22, 14].

While SRResNet [14] successfully solved those time and memory issue with good performance, it simply employs the ResNet architecture from He et al. [9] without much modification. However, original ResNet was proposed to solve higher-level computer vision problems such as image classification and detection. Therefore, applying ResNet architecture directly to low-level vision problems like super-resolution can be suboptimal.

To solve these problems, based on the SRResNet architecture, we first optimize it by analyzing and removing unnecessary modules to simplify the network architecture. Training a network becomes nontrivial when the model is complex. Thus, we train the network with appropriate loss function and careful model modification upon training. We experimentally show that the modified scheme produces better results.

Second, we investigate the model training method that transfers knowledge from a model trained at other scales. To utilize scale-independent information during training, we train high-scale models from pre-trained low-scale models. Furthermore, we propose a new multi-scale architecture that shares most of the parameters across different scales. The proposed multi-scale model uses significantly fewer parameters compared with multiple single-scale models but shows comparable performance.

We evaluate our models on the standard benchmark datasets and on a newly provided DIV2K dataset. The proposed single- and multi-scale super-resolution networks show the state-of-the-art performances on all datasets in terms of PSNR and SSIM. Our methods ranked first and second, respectively, in the NTIRE 2017 Super-Resolution Challenge [26].

2. Related Works

To solve the super-resolution problem, early approaches use interpolation techniques based on sampling theory [1, 15, 34]. However, those methods exhibit limitations in predicting detailed, realistic textures. Previous studies [25, 23] adopted natural image statistics to the problem to reconstruct better high-resolution images.

Advanced works aim to learn mapping functions between I^{LR} and I^{HR} image pairs. Those learning methods rely on techniques ranging from neighbor embedding [3, 2, 7, 21] to sparse coding [31, 32, 27, 33]. Yang et al. [30] introduced another approach that clusters the patch spaces and learns the corresponding functions. Some approaches utilize image self-similarities to avoid using external databases [8, 6, 29], and increase the size of the limited internal dictionary by geometric transformation of patches [10].

Recently, the powerful capability of deep neural networks has led to dramatic improvements in SR. Since Dong et al. [4, 5] first proposed a deep learning-based SR method, various CNN architectures have been studied for SR. Kim et al. [11, 12] first introduced the residual network for training much deeper network architectures and achieved superior performance. In particular, they showed that skip-connection and recursive convolution alleviate the burden of carrying identity information in the super-resolution network. Similarly to [20], Mao et al. [16] tackled the general image restoration problem with encoder-decoder networks and symmetric skip connections. In [16], they argue that those nested skip connections provide fast and improved convergence.

In many deep learning based super-resolution algorithms, an input image is upsampled via bicubic interpolation before they fed into the network [4, 11, 12]. Rather than using an interpolated image as an input, training upsampling modules at the very end of the network is also possible as shown in [5, 22, 14]. By doing so, one can reduce much of computations without losing model capacity because the size of features decreases. However, those kinds of approaches have one disadvantage: They cannot deal with the multi-scale problem in a single framework as in VDSR [11]. In this work, we resolve the dilemma of multi-scale training and computational efficiency. We not only exploit the inter-relation of learned feature for each scale but also propose a new multi-scale model that efficiently reconstructs high-resolution images for various scales. Furthermore, we develop an appropriate training method that uses multiple scales for both single- and multi-scale models.

Several studies also have focused on the loss functions to better train network models. Mean squared error (MSE) or L2 loss is the most widely used loss function for general image restoration and is also major performance measure (PSNR) for those problems. However, Zhao et al. [35] reported that training with L2 loss does not guarantee better performance compared to other loss functions in terms of PSNR and SSIM. In their experiments, a network trained with L1 achieved improved performance compared with the network trained with L2.

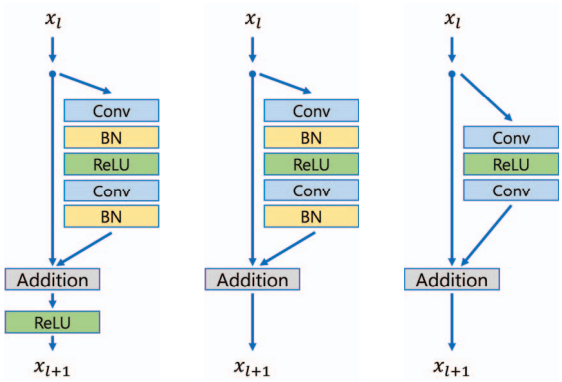
3. Proposed Methods

In this section, we describe proposed model architectures. We first analyze recently published super-resolution network and suggest an enhanced version of the residual

network architecture with the simpler structure. We show that our network outperforms the original ones while exhibiting improved computational efficiency. In the following sections, we suggest a single-scale architecture (EDSR) that handles a specific super-resolution scale and a multi-scale architecture (MDSR) that reconstructs various scales of high-resolution images in a single model.

3.1. Residual blocks

Recently, residual networks [11, 9, 14] exhibit excellent performance in computer vision problems from the low-level to high-level tasks. Although Ledig et al. [14] successfully applied the ResNet architecture to the super-resolution problem with SRResNet, we further improve the performance by employing better ResNet structure.



(a) Original (b) SRResNet (c) Proposed

Figure 2: Comparison of residual blocks in original ResNet, SRResNet, and ours.

In Fig. 2, we compare the building blocks of each network model from original ResNet [9], SRResNet [14], and our proposed networks. We remove the batch normalization layers from our network as Nah et al.[19] presented in their image deblurring work. Since batch normalization layers normalize the features, they get rid of range flexibility from networks by normalizing the features, it is better to remove them. We experimentally show that this simple modification increases the performance substantially as detailed in Sec. 4.

Furthermore, GPU memory usage is also sufficiently reduced since the batch normalization layers consume the same amount of memory as the preceding convolutional layers. Our baseline model without batch normalization layer saves approximately 40% of memory usage during training, compared to SRResNet. Consequently, we can build up a larger model that has better performance than conventional ResNet structure under limited computational resources.

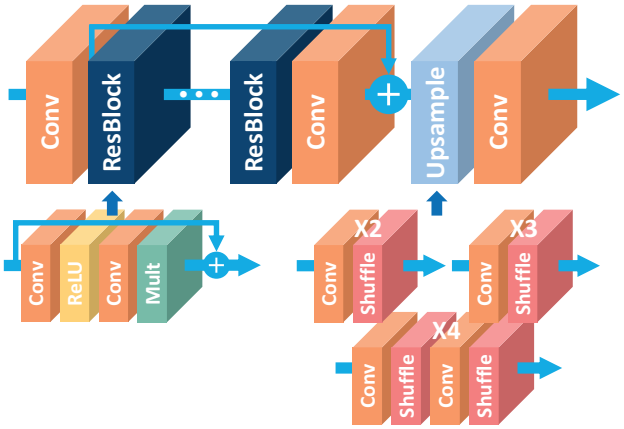


Figure 3: The architecture of the proposed single-scale SR network (EDSR).

3.2. Single-scale model

The simplest way to enhance the performance of the network model is to increase the number of parameters. In the convolutional neural network, model performance can be enhanced by stacking many layers or by increasing the number of filters. General CNN architecture with depth (the number of layers) B and width (the number of feature channels) F occupies roughly $O(BF)$ memory with $O(BF^2)$ parameters. Therefore, increasing F instead of B can maximize the model capacity when considering limited computational resources.

However, we found that increasing the number of feature maps above a certain level would make the training procedure numerically unstable. A similar phenomenon was reported by Szegedy et al. [24]. We resolve this issue by adopting the residual scaling [24] with factor 0.1. In each residual block, constant scaling layers are placed after the last convolution layers. These modules stabilize the training procedure greatly when using a large number of filters. In the test phase, this layer can be integrated into the previous convolution layer for the computational efficiency.

We construct our **baseline (single-scale)** model with our proposed residual blocks in Fig. 2. The structure is similar to SRResNet [14], but our model does not have ReLU activation layers outside the residual blocks. Also, our baseline model does not have residual scaling layers because we use only 64 feature maps for each convolution layer. In our final single-scale model (**EDSR**), we expand the baseline model by setting $B = 32$, $F = 256$ with a scaling factor 0.1. The model architecture is displayed in Fig. 3.

When training our model for upsampling factor $\times 3$ and $\times 4$, we initialize the model parameters with pre-trained $\times 2$ network. This pre-training strategy accelerates the training and improves the final performance as clearly demonstrated in Fig. 4. For upscaling $\times 4$, if we use a pre-trained scale $\times 2$

model (blue line), the training converges much faster than the one started from random initialization (green line).

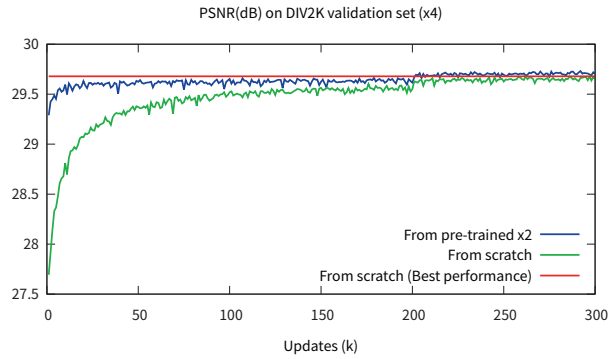


Figure 4: Effect of using pre-trained $\times 2$ network for $\times 4$ model (EDSR). The red line indicates the best performance of green line. 10 images are used for validation during training.

3.3. Multi-scale model

From the observation in Fig. 4, we conclude that super-resolution at multiple scales is inter-related tasks. We further explore this idea by building a multi-scale architecture that takes the advantage of inter-scale correlation as VDSR [11] does. We design our **baseline (multi-scale)** models to have a single main branch with $B = 16$ residual blocks so that most of the parameters are shared across different scales as shown in Fig. 5.

In our multi-scale architecture, we introduce scale-specific processing modules to handle the super-resolution at multiple scales. First, pre-processing modules are located at the head of networks to reduce the variance from input images of different scales. Each of pre-processing module consists of two residual blocks with 5×5 kernels. By adopting larger kernels for pre-processing modules, we can keep the scale-specific part shallow while the larger receptive field is covered in early stages of networks. At the end of the multi-scale model, scale-specific upsampling modules are located in parallel to handle multi-scale reconstruction. The architecture of the upsampling modules is similar to those of single-scale models described in the previous section.

We construct our final multi-scale model (**MDSR**) with $B = 80$ and $F = 64$. While our single-scale baseline models for 3 different scales have about 1.5M parameters each, totaling 4.5M, our baseline multi-scale model has only 3.2 million parameters. Nevertheless, the multi-scale model exhibits comparable performance as the single-scale models. Furthermore, our multi-scale model is scalable in terms of depth. Although our final MDSR has approximately 5 times more depth compared to the baseline multi-scale model, only 2.5 times more parameters are required, as the residual blocks are lighter than scale-specific parts. Note that MDSR also shows the comparable performance to the scale-specific EDSRs. The detailed performance comparison of our proposed models is presented in Table 2 and 3.

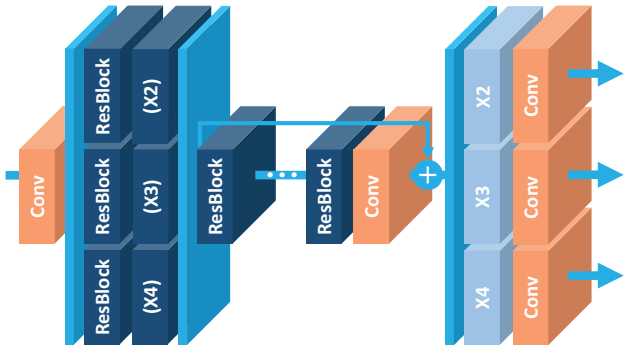


Figure 5: The architecture of the proposed multi-scale SR network (MDSR).

Options	SRResNet [14] (reproduced)	Baseline (Single / Multi)	EDSR	MDSR
# Residual blocks	16	16	32	80
# Filters	64	64	256	64
# Parameters	1.5M	1.5M / 3.2M	43M	8.0M
Residual scaling	-	-	0.1	-
Use BN	Yes	No	No	No
Loss function	L2	L1	L1	L1

Table 1: Model specifications.

4. Experiments

4.1. Datasets

DIV2K dataset [26] is a newly proposed high-quality (2K resolution) image dataset for image restoration tasks. The DIV2K dataset consists of 800 training images, 100 validation images, and 100 test images. As the test dataset ground truth is not released, we report and compare the performances on the validation dataset. We also compare the performance on four standard benchmark datasets: Set5 [2], Set14 [33], B100 [17], and Urban100 [10].

4.2. Training Details

For training, we use the RGB input patches of size 48×48 from LR image with the corresponding HR patches. We augment the training data with random horizontal flips and 90 rotations. We pre-process all the images by subtracting the mean RGB value of the DIV2K dataset. We train

Scale	SRResNet (L2 loss)	SRResNet (L1 loss)	Our baseline (Single-scale)	Our baseline (Multi-scale)	EDSR (Ours)	MDSR (Ours)	**EDSR+ (Ours)**	**MDSR+ (Ours)**
$\times 2$	34.40 / 0.9662	34.44 / 0.9665	34.55 / 0.9671	34.60 / 0.9673	35.03 / 0.9695	34.96 / 0.9692	35.12 / 0.9699	35.05 / 0.9696
$\times 3$	30.82 / 0.9288	30.85 / 0.9292	30.90 / 0.9298	30.91 / 0.9298	31.26 / 0.9340	31.25 / 0.9338	31.39 / 0.9351	31.36 / 0.9346
$\times 4$	28.92 / 0.8960	28.92 / 0.8961	28.94 / 0.8963	28.95 / 0.8962	29.25 / 0.9017	29.26 / 0.9016	29.38 / 0.9032	29.36 / 0.9029

Table 2: Performance comparison between architectures on the DIV2K validation set (PSNR(dB) / SSIM). Red indicates the best performance and blue indicates the second best. EDSR+ and MDSR+ denote self-ensemble versions of EDSR and MDSR.

our model with ADAM optimizer [13] by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We set minibatch size as 16. The learning rate is initialized as 10^{-4} and halved at every 2×10^5 minibatch updates.

For the single-scale models (EDSR), we train the networks as described in Sec. 3.2. The $\times 2$ model is trained from scratch. After the model converges, we use it as a pre-trained network for other scales.

At each update of training a multi-scale model (MDSR), we construct the minibatch with a randomly selected scale among $\times 2$, $\times 3$ and $\times 4$. Only the modules that correspond to the selected scale are enabled and updated. Hence, scale-specific residual blocks and upsampling modules that correspond to different scales other than the selected one are not enabled nor updated.

We train our networks using L1 loss instead of L2. Minimizing L2 is generally preferred since it maximizes the PSNR. However, based on a series of experiments we empirically found that L1 loss provides better convergence than L2. The evaluation of this comparison is provided in Sec. 4.4

We implemented the proposed networks with the Torch7 framework and trained them using NVIDIA Titan X GPUs. It takes 8 days and 4 days to train EDSR and MDSR, respectively. The source code is publicly available online.[1]

4.3. Geometric Self-ensemble

In order to maximize the potential performance of our model, we adopt the self-ensemble strategy similarly to [28]. During the test time, we flip and rotate the input image I^{LR} to generate seven augmented inputs $I^{LR}_{n,i} = T_i\left(I^{LR}_n\right)$ for each sample, where T_i represents the 8 geometric transformations including indentity. With those augmented low-resolution images, we generate corresponding super-resolved images $\left\{I^{SR}_{n,1}, \cdots, I^{SR}_{n,8}\right\}$ using the networks. We then apply inverse transform to those output images to get the original geometry $\tilde{I}^{SR}_{n,i} = T_i^{-1}\left(I^{SR}_{n,i}\right)$. Finally, we average the transformed outputs all together to make the self-ensemble result as follows. $I^{SR}_n = \frac{1}{8}\sum_{i=1}^{8} \tilde{I}^{SR}_{n,i}$.

This self-ensemble method has an advantage over other ensembles as it does not require additional training of separate models. It is beneficial especially when the model size or training time matters. Although self-ensemble strategy keeps the total number of parameters same, we notice that it gives approximately same performance gain compared to conventional model ensemble method that requires individually trained models. We denote the methods using self-ensemble by adding '+' postfix to the method name; i.e. EDSR+/MDSR+. Note that geometric self-ensemble is valid only for symmetric downsampling methods such as bicubic downsampling.

4.4. Evaluation on DIV2K Dataset

We test our proposed networks on the DIV2K dataset. Starting from the SRResNet, we gradually change various settings to perform ablation tests. We train SRResNet [14] on our own. [2] [3] First, we change the loss function from L2 to L1, and then the network architecture is reformed as described in the previous section and summarized in Table 1.

We train all those models with 3×10^5 updates in this experiment. Evaluation is conducted on the 10 images of DIV2K validation set, with PSNR and SSIM criteria. For the evaluation, we use full RGB channels and ignore the (6 + scale) pixels from the border.

Table 2 presents the quantitative results. SRResNet trained with L1 gives slightly better results than the original one trained with L2 for all scale factors. Modifications of the network give an even bigger margin of improvements. The last 2 columns of Table 2 show significant performance gains of our final bigger models, EDSR+ and MDSR+ with the geometric self-ensemble technique Note that our models require much less GPU memory since they do not have batch normalization layers.

[1] https://github.com/LimBee/NTIRE2017

[2] We confirmed our reproduction is correct by getting comparable results in an individual experiment, using the same settings of the paper [14]. In our experiments, however, it became slightly different to match the settings of our baseline model training. See our codes at https://github.com/LimBee/NTIRE2017.

[3] We used the original paper (https://arxiv.org/abs/1609.04802v3) as a reference.

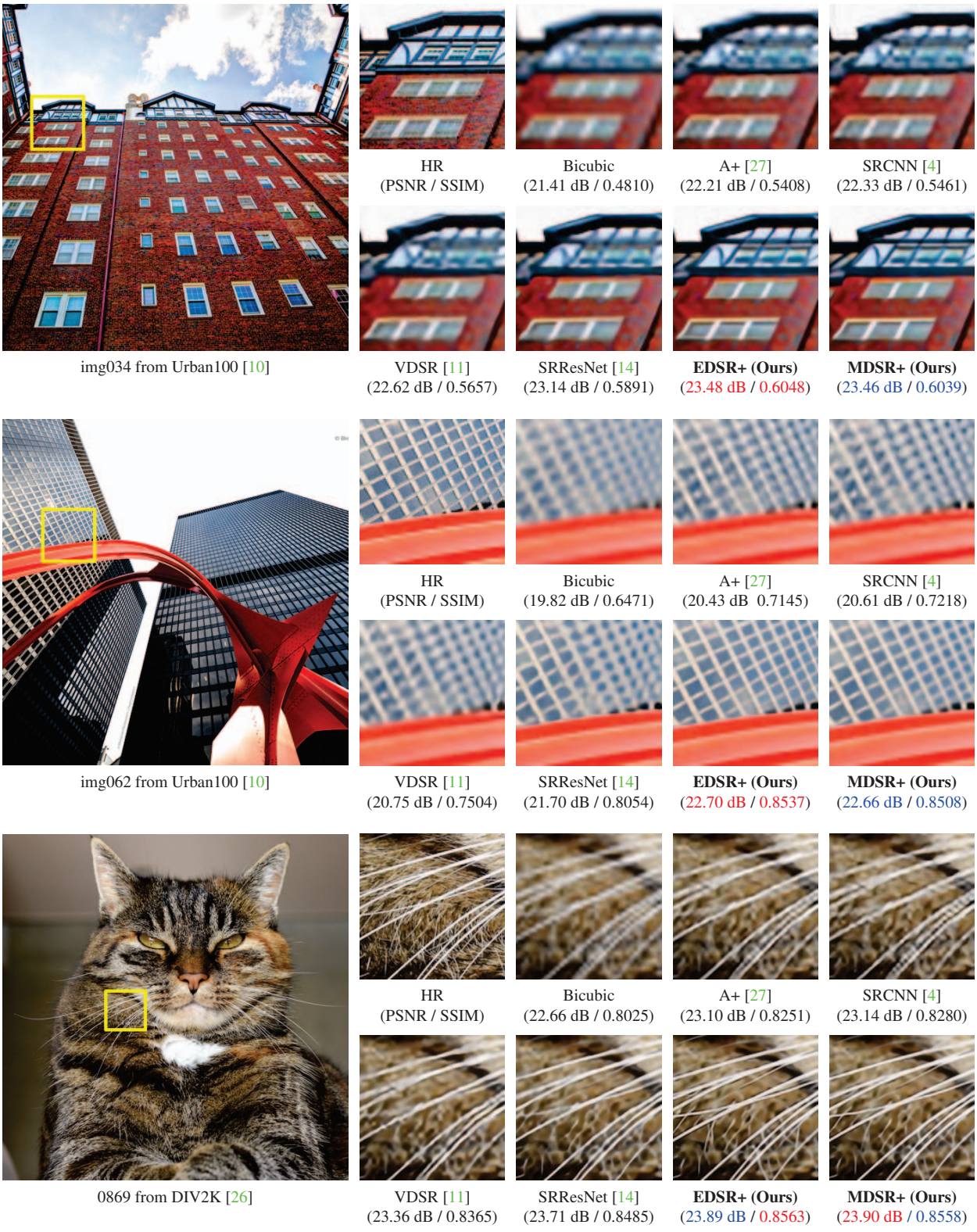


Figure 6: Qualitative comparison of our models with other works on $\times 4$ super-resolution.

Dataset	Scale	Bicubic	A+ [27]	SRCNN [4]	VDSR [11]	SRResNet [14]	EDSR (Ours)	MDSR (Ours)	**EDSR+ (Ours)**	**MDSR+ (Ours)**
Set5	×2	33.66 / 0.9299	36.54 / 0.9544	36.66 / 0.9542	37.53 / 0.9587	- / -	38.11 / 0.9601	38.11 / 0.9602	38.20 / 0.9606	38.17 / 0.9605
	×3	30.39 / 0.8682	32.58 / 0.9088	32.75 / 0.9090	33.66 / 0.9213	- / -	34.65 / 0.9282	34.66 / 0.9280	34.76 / 0.9290	34.77 / 0.9288
	×4	28.42 / 0.8104	30.28 / 0.8603	30.48 / 0.8628	31.35 / 0.8838	32.05 / 0.8910	32.46 / 0.8968	32.50 / 0.8973	32.62 / 0.8984	32.60 / 0.8982
Set14	×2	30.24 / 0.8688	32.28 / 0.9056	32.42 / 0.9063	33.03 / 0.9124	- / -	33.92 / 0.9195	33.85 / 0.9198	34.02 / 0.9204	33.92 / 0.9203
	×3	27.55 / 0.7742	29.13 / 0.8188	29.28 / 0.8209	29.77 / 0.8314	- / -	30.52 / 0.8462	30.44 / 0.8452	30.66 / 0.8481	30.53 / 0.8465
	×4	26.00 / 0.7027	27.32 / 0.7491	27.49 / 0.7503	28.01 / 0.7674	28.53 / 0.7804	28.80 / 0.7876	28.72 / 0.7857	28.94 / 0.7901	28.82 / 0.7876
B100	×2	29.56 / 0.8431	31.21 / 0.8863	31.36 / 0.8879	31.90 / 0.8960	- / -	32.32 / 0.9013	32.29 / 0.9007	32.37 / 0.9018	32.34 / 0.9014
	×3	27.21 / 0.7385	28.29 / 0.7835	28.41 / 0.7863	28.82 / 0.7976	- / -	29.25 / 0.8093	29.25 / 0.8091	29.32 / 0.8104	29.30 / 0.8101
	×4	25.96 / 0.6675	26.82 / 0.7087	26.90 / 0.7101	27.29 / 0.7251	27.57 / 0.7354	27.71 / 0.7420	27.72 / 0.7418	27.79 / 0.7437	27.78 / 0.7425
Urban100	×2	26.88 / 0.8403	29.20 / 0.8938	29.50 / 0.8946	30.76 / 0.9140	- / -	32.93 / 0.9351	32.84 / 0.9347	33.10 / 0.9363	33.03 / 0.9362
	×3	24.46 / 0.7349	26.03 / 0.7973	26.24 / 0.7989	27.14 / 0.8279	- / -	28.80 / 0.8653	28.79 / 0.8655	29.02 / 0.8685	28.99 / 0.8683
	×4	23.14 / 0.6577	24.32 / 0.7183	24.52 / 0.7221	25.18 / 0.7524	26.07 / 0.7839	26.64 / 0.8033	26.67 / 0.8041	26.86 / 0.8080	26.86 / 0.8082
DIV2K validation	×2	31.01 / 0.9393	32.89 / 0.9570	33.05 / 0.9581	33.66 / 0.9625	- / -	35.03 / 0.9695	34.96 / 0.9692	35.12 / 0.9699	35.05 / 0.9696
	×3	28.22 / 0.8906	29.50 / 0.9116	29.64 / 0.9138	30.09 / 0.9208	- / -	31.26 / 0.9340	31.25 / 0.9338	31.39 / 0.9351	31.36 / 0.9346
	×4	26.66 / 0.8521	27.70 / 0.8736	27.78 / 0.8753	28.17 / 0.8841	- / -	29.25 / 0.9017	29.26 / 0.9016	29.38 / 0.9032	29.36 / 0.9029

Table 3: Public benchmark test results and DIV2K validation results (PSNR(dB) / SSIM). Red indicates the best performance and blue indicates the second best. Note that DIV2K validation results are acquired from published demo codes.

4.5. Benchmark Results

We provide the quantitative evaluation results of our final models (EDSR+, MDSR+) on public benchmark datasets in Table 3. The evaluation of the self-ensemble is also provided in the last two columns. We trained our models using 10^6 updates with batch size 16. We keep the other settings same as the baseline models. We compare our models with the state-of-the-art methods including A+ [27], SRCNN [4], VDSR [11], and SRResNet [14]. For comparison, we measure PSNR and SSIM on the y channel and ignore the same amount of pixels as scales from the border. We used MATLAB [18] functions for evaluation. Comparative results on DVI2K dataset are also provided. Our models exhibit a significant improvement compared to the other methods. The gaps further increase after performing self-ensemble. We also present the qualitative results in Fig. 6. The proposed models successfully reconstruct the detailed textures and edges in the HR images and exhibit better-looking SR outputs compared with the previous works.

5. NTIRE2017 SR Challenge

This work is initially proposed for the purpose of participating in the NTIRE2017 Super-Resolution Challenge [26]. The challenge aims to develop a single image super-resolution system with the highest PSNR.

In the challenge, there exist two tracks for different degraders (bicubic, unknown) with three downsample scales $(\times 2, 3, 4)$ each. Input images for the unknown track are not only downscaled but also suffer from severe blurring. Therefore, more robust mechanisms are required to deal with the second track. We submitted our two SR models (EDSR and MDSR) for each competition and prove that our algorithms are very robust to different downsampling conditions. Some results of our algorithms on the unknown downsampling track are illustrated in Fig. 7. Our methods successfully reconstruct high-resolution images from severely degraded input images. Our proposed EDSR+ and MDSR+ won the first and second places, respectively, with outstanding performances as shown in Table 4.

6. Conclusion

In this paper, we proposed an enhanced super-resolution algorithm. By removing unnecessary modules from conventional ResNet architecture, we achieve improved results while making our model compact. We also employ residual scaling techniques to stably train large models. Our proposed singe-scale model surpasses current models and achieves the state-of-the-art performance.

Furthermore, we develop a multi-scale super-resolution network to reduce the model size and training time. With scale-dependent modules and shared main network, our multi-scale model can effectively deal with various scales of super-resolution in a unified framework. While the multi-scale model remains compact compared with a set of single-scale models, it shows comparable performance to the single-scale SR model.

Our proposed single-scale and multi-scale models have achieved the top ranks in both the standard benchmark datasets and the DIV2K dataset.

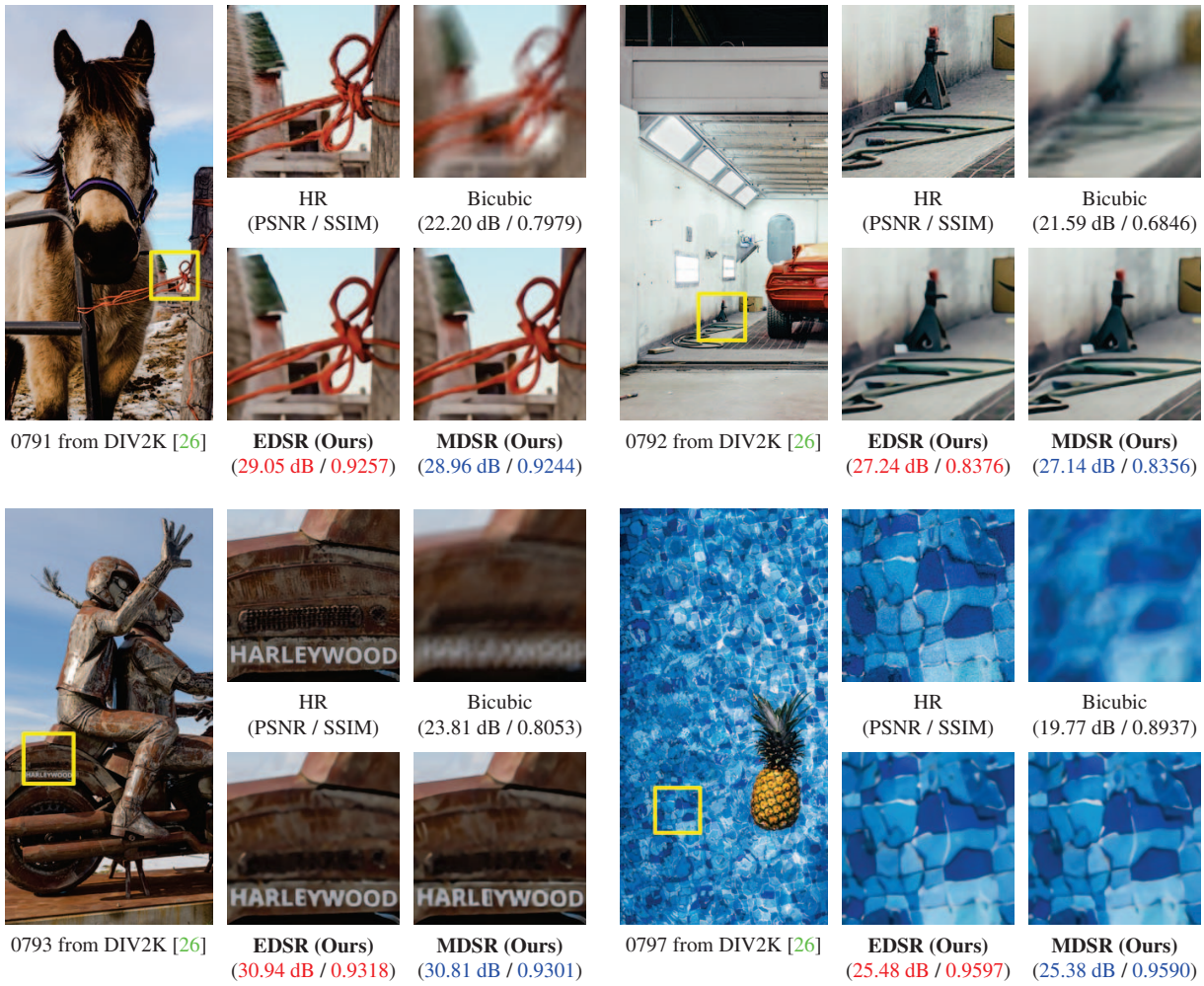


Figure 7: Our NTIRE2017 Super-Resolution Challenge results on unknown downscaling $\times 4$ category. In the challenge, we excluded images from 0791 to 0800 from training for validation. We did not use geometric self-ensemble for unknown downscaling category.

	Track1: bicubic downscailing						Track2: unknown downscailing					
	$\times 2$		$\times 3$		$\times 4$		$\times 2$		$\times 3$		$\times 4$	
Method	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
EDSR+ (Ours)	34.93	0.948	31.13	0.889	29.09	0.837	34.00	0.934	30.78	0.881	28.77	0.826
MDSR+ (Ours)	34.83	0.947	31.04	0.888	29.04	0.836	33.86	0.932	30.67	0.879	28.62	0.821
3rd method	34.47	0.944	30.77	0.882	28.82	0.830	33.67	0.930	30.51	0.876	28.54	0.819
4th method	34.66	0.946	30.83	0.884	28.83	0.830	32.92	0.921	30.31	0.871	28.14	0.807
5th method	34.29	0.948	30.52	0.889	28.55	0.752	-	-	-	-	-	-

Table 4: Performance of our methods on the test dataset of NTIRE2017 Super-Resolution Challenge [26]. The results of top 5 methods are displayed for two tracks and six categories. Red indicates the best performance and blue indicates the second best.

References

[1] J. Allebach and P. W. Wong. Edge-directed interpolation. In *ICIP 1996*. 2

[2] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC 2012*. 2, 4

[3] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *CVPR 2004*. 2

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV 2014*. 2, 6, 7

[5] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV 2016*. 2

[6] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics (TOG)*, 30(2):12, 2011. 2

[7] X. Gao, K. Zhang, D. Tao, and X. Li. Image super-resolution with sparse neighbor embedding. *IEEE Transactions on Image Processing*, 21(7):3194–3205, 2012. 2

[8] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV 2009*. 2

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR 2016*. 3

[10] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR 2015*. 2, 4, 6

[11] J. Kim, J. Kwon Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR 2016*. 1, 2, 3, 4, 6, 7

[12] J. Kim, J. Kwon Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR 2016*. 1, 2

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR 2014*. 4

[14] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv:1609.04802*, 2016. 1, 2, 3, 4, 5, 6, 7

[15] X. Li and M. T. Orchard. New edge-directed interpolation. *IEEE Transactions on Image Processing*, 10(10):1521–1527, 2001. 2

[16] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS 2016*. 2

[17] 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 *ICCV 2001*. 4

[18] MATLAB. *version 9.1.0 (R2016b)*. The MathWorks Inc., Natick, Massachusetts, 2016. 7

[19] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. *arXiv:1612.02177*, 2016. 3

[20] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI 2015*. 2

[21] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 2

[22] 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 *CVPR 2016*. 2

[23] J. Sun, Z. Xu, and H.-Y. Shum. Image super-resolution using gradient profile prior. In *CVPR 2008*. 2

[24] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv:1602.07261*, 2016. 3

[25] Y.-W. Tai, S. Liu, M. S. Brown, and S. Lin. Super resolution using edge prior and single image detail synthesis. In *CVPR 2010*. 2

[26] 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 *CVPR 2017 Workshops*. 1, 2, 4, 6, 7, 8

[27] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV 2014*. 2, 6, 7

[28] R. Timofte, R. Rothe, and L. Van Gool. Seven ways to improve example-based single image super resolution. In *CVPR 2016*. 5

[29] Z. Wang, Y. Yang, Z. Wang, S. Chang, J. Yang, and T. S. Huang. Learning super-resolution jointly from external and internal examples. *IEEE Transactions on Image Processing*, 24(11):4359–4371, 2015. 2

[30] C.-Y. Yang and M.-H. Yang. Fast direct super-resolution by simple functions. In *ICCV 2013*. 2

[31] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *IEEE Transactions on Image Processing*, 21(8):3467–3478, 2012. 2

[32] 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. 2

[33] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Proceedings of the International Conference on Curves and Surfaces*, 2010. 2, 4

[34] L. Zhang and X. Wu. An edge-guided image interpolation algorithm via directional filtering and data fusion. *IEEE Transactions on Image Processing*, 15(8):2226–2238, 2006. 2

[35] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for neural networks for image processing. *arXiv:1511.08861*, 2015. 2