Deep Networks for Single Image Super-Resolution with Multi-context Fusion

Zheng Hui, Xiumei Wang(✉), and Xinbo Gao

School of Electronic Engineering, Xidian University, Xi'an 710071, China
zheng_hui@aliyun.com, wangxm@xidian.edu.cn, xbgao@mail.xidian.edu.cn

Abstract. Deep convolutional neural networks have been successfully applied to image super resolution. In this paper, we propose a multi-context fusion learning based super resolution model to exploit context information on both smaller image regions and larger image regions for SR. To speed up execution time, our method directly takes the low-resolution image (not interpolation version) as input on both training and testing processes and combines the residual network at the same time. The proposed model is extensively evaluated and compared with the state-of-the-art SR methods and experimental results demonstrate its performance in speed and accuracy.

Keywords: Convolutional neural networks · Super-resolution
Multi-context fusion

1 Introduction

Aim to recover a high-resolution (HR) image from the corresponding low-resolution (LR) image, single image super-resolution (SR) has been utilized in the area of computer vision for several decades. Typical image SR methods can be roughly categorized into three kinds, i.e., interpolation-based, reconstruction-based, and learning-based [1].

Recently, learning-based SR methods using convolutional neural network (CNN) have demonstrated notable progress. Dong *et al.* first proposed a CNN-based SR method known as SRCNN, which learns a mapping from LR to HR in an end-to-end manner and shows the state-of-art performance in accuracy and visual. A neural network that closely mimics the sparse coding approach for image SR is proposed by Wang et al. [2]. Kim *et al.* proposed a very deep neural network with residual architecture to exploit contextual information over large image regions [3].

Although the CNN-based models mentioned above have achieved good performance, there are two main issues should not be neglected. First, most of existing methods upscale a single low-resolution image to the desired size using bicubic interpolation before applying the network for prediction. This pre-processing step increases unnecessary computational cost and often results in visible reconstruction artifacts. Several approaches are observed to perform better which accelerate SRCNN directly by learning from the LR image and embedding the resolution change into the network [4]. These method, however, use relatively small networks and cannot learn complicated mappings well due to the limited network capacity. In addition, most methods rely on the context information of small image regions. These ways don't make full use of the information of large image regions.

To solve the aforementioned issues, we propose a Multi-context Fusion Super-Resolution Network (MFSR) based on a cascade of convolutional neural networks (CNNs) and a residual network. Note that very deep CNN model would increase the model capacity, but it would introduce more parameters which is infeasible on the scene requiring the higher speed. Thus, an architecture with trade-off between accuracy and speed is essential for practical applications. In this paper, we adopt the suitable network width (namely the number of filters in a layer) to reduce the amounts of parameters. Experiments prove that the proposed model can achieve both state-of-the-art PSNR and SSIM results and visually pleasant results.

2 Related Work

Lately, many convolutional neural network based SR methods have been proposed in the papers. So, we focus on recent convolutional neural networks based SR approaches.

2.1 Convolutional Neural Network for Image Super-Resolution

The SRCNN [1] aims at learning an end-to-end mapping, which takes the low-resolution image Y as input and directly outputs the high-resolution one $F(Y)$. This model contains three layers: patch extraction and representation, non-linear mapping and reconstruction. The VDSR network [3] demonstrates significant improvement over SRCNN [1] by increasing the network depth from 3 to 20 convolutional layers. To speed-up the training, VDSR suggests the residual-learning CNN and uses high learning rates. To achieve real-time performance, the FSR-CNN network [4] takes the LR image as input and enlarges feature maps by using transposed convolution.

2.2 Residue Learning

It is pointed by [3] that enabling residue learning during network training results in a faster convergence as well as a better performance in the final result. Kim *et al.* point out carrying the input to the end is conceptually similar to what an auto-encoder does. In this way, the network concentrates on the learning of residual image, so that the convergence rate is significantly decreased. To facilitate residue learning in our network, we up-sample the input LR image with bicubic interpolation algorithm, then add it to the output of the transposed convolutional layer.

Our approach builds upon existing CNN-based SR algorithms with a main difference. We joint residual learning and multi-context fusion learning by using convolutional and transposed convolutional layers.

3 Proposed Method

In this section, we describe the design methodology of the proposed multi-context fusion learning network.

3.1 Network Architecture

We propose to construct the network by using a series of convolutional layers, as shown in Fig. 1. Our model takes an LR image as input (rather than interpolated version of the LR image) and predicts a residual image. The proposed model has two parts, feature extraction and image reconstruction.

Feature Extraction. We use n cascaded containers, one convolutional layer and one transposed convolutional layer upsampling the extracted features. Each

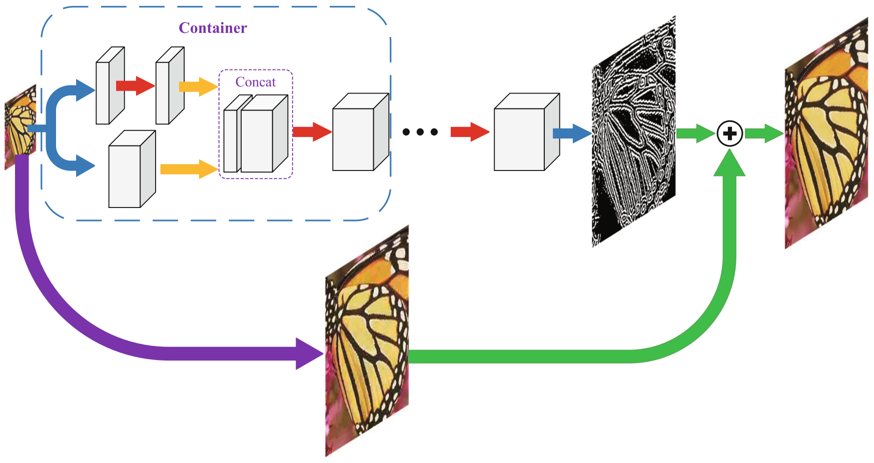


Fig. 1. The overview of our proposed method. Red arrows indicate convolutional layers. The blue straight arrow indicates transposed convolutional layer and the blue curved arrow indicates the combination of the split operation and convolutional operation, namely split operation is performed first and then convolutional operation is deployed. The purple arrow denotes bicubic interpolation operator and the green arrow denotes element-wise addition operator. Orange arrows and 'Concat box' represent data flow from orange arrows to 'Concat box' to accomplish feature maps fit together in channel dimension. We cascade n containers (broken blue box) repeatedly. A low-resolution (LR) image goes through our network and transforms into a high-resolution (HR) image. (Color figure online)

container has two parallel paths, which are used as multi-context fusion features extractor. One path contains two cascaded convolutional layers and each convolutional layer has 16 filters of spatial sizes 3×3, the other path has one convolutional layer with 32 filters of spatial sizes 3×3. Two 3×3 layers bring receptive field with the same size as one 5×5 layer. Each of our containers uses receptive fields of different sizes for taking context information on the larger image regions and the smaller image regions into account simultaneously and can be regarded as local context fusion unit. Here, sub-networks in each container adopt two different depths which means to combine contextual feature information on both shallow sub-network and deep sub-network. And then, one 3×3 layer with 48 filters merges these extracted feature maps and takes the results as the next container's input. And a convolutional layer with 64 filters of size 3×3 is followed by the last container mainly increase the dimension of previous extracted features. The last layer on this stage adopts one transposed convolutional layer with 1 filter of size 9×9 to upsample and aggregate the previous generated feature maps. Then the output is the predicted residual image.

Image Reconstruction. On this stage, the input coarse image is upsampled by the customized bicubic upscaling layer. The upsampled image is then combined (using element-wise summation) with the predicted residual image from the feature extraction stage to produce a high-resolution output image.

3.2 Loss Function

Mean square error (MSE) is employed as the loss function to train the network, and our optimization objective can be expressed as

$$\min_{\Theta} \sum_{i} \left\| MFSR\left(I_y^{(i)}; \Theta\right) - I_x^{(i)} \right\|_2^2, \tag{1}$$

where $I_y^{(i)}$ and $I_x^{(i)}$ are the i-th pair of LR/HR training data, and $MFSR\,(I_y; \Theta)$ denotes the HR image for I_y predicted using the MFSR model with parameter set Θ. All the parameters are optimized through the standard back-propagation algorithm.

4 Experiments

In this section, we first explain the implementation details in the experiments and then compare the MFSR with state-of-the-arts on four benchmark datasets to demonstrate the effectiveness of our proposed method.

4.1 Implementation Details

In the proposed model, we initialize the convolutional filters using the method of He *et al.* [5]. The size of the transposed convolutional filter is 9×9 and the weight is initialized from bilinear interpolation kernel. For the activation function after each convolution layer, we suggest the use of Parametric Rectified Linear Unit (PReLU) [5] instead of the commonly-used Rectified Linear Unit (ReLU). We pad zeros around the boundaries before applying convolution to keep the size of all feature maps. As the training is implemented with the *caffe* package [6], the transposed convolution filter will output a feature map with $(s-1)$-pixel cut on the border (s is the stride of kernel namely magnification). As to the number of the containers, we set $n = 7$ to achieve the trade-off between performance and the execution time.

We use 91 images from Yang *et al.* [7] and 200 images from Berkeley Segmentation Dataset [8] as our training data. The optimization is conducted by the mini-batch stochastic gradient descent method with a batch size of 64, momentum of 0.9, and weight decay of $1e-4$. The learning rate is initially set to $1e-4$ and fixed during the whole training phase. In addition, data augmentation (rotation, scaling and flipping) is used.

Table 1. Quantitative evaluation of state-of-art SR algorithms: average PSNR/SSIM for scale 2×, 3× and 4×. **Red** text indicates the best and **blue** text indicates the second best performance.

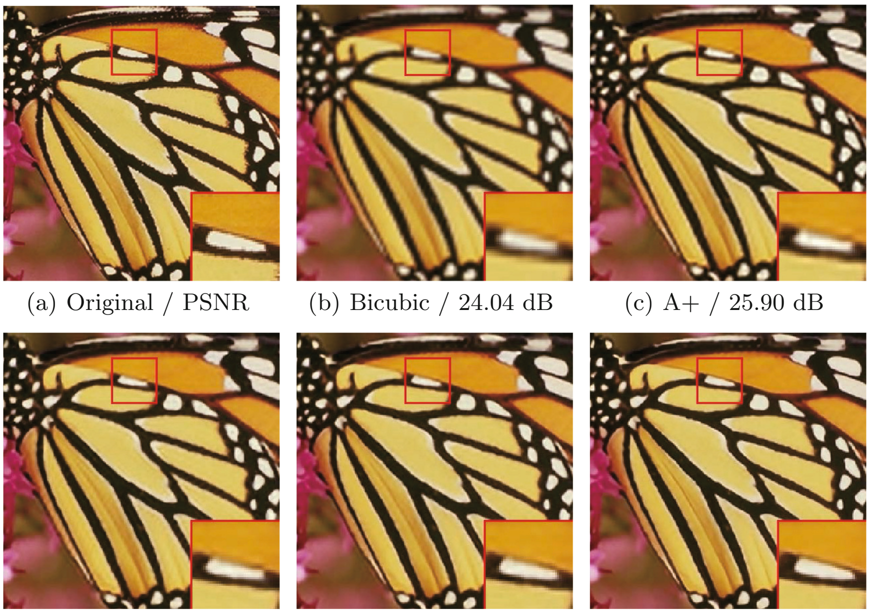
Algorithm	Scale	Set5	Set14	BSD100	Urban100
		PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Bicubic	2	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	26.88 / 0.8403
A+	2	36.54 / 0.9544	32.28 / 0.9056	31.21 / 0.8863	29.20 / 0.8938
SRCNN	2	36.66 / 0.9542	32.45 / 0.9067	31.36 / 0.8879	29.50 / 0.8946
SelfExSR	2	36.49 / 0.9537	32.22 / 0.9034	31.18 / 0.8855	29.54 / 0.8967
RFL	2	36.54 / 0.9537	32.26 / 0.9040	31.16 / 0.8840	29.11 / 0.8904
FSRCNN	2	37.00 / 0.9558	32.63 / 0.9088	31.50 / 0.8906	29.85 / 0.9009
MFSR	2	**37.71 / 0.9593**	**33.13 / 0.9132**	**31.94 / 0.8965**	**30.80 / 0.9143**
Bicubic	3	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	24.46 / 0.7349
A+	3	32.59 / 0.9088	29.13 / 0.8188	28.29 / 0.7835	26.03 / 0.7973
SRCNN	3	32.75 / 0.9090	29.28 / 0.8209	28.41 / 0.7863	26.24 / 0.7989
SelfExSR	3	32.58 / 0.9093	29.16 / 0.8196	28.29 / 0.7840	26.44 / 0.8088
RFL	3	32.43 / 0.9057	29.05 / 0.8164	28.22 / 0.7806	25.86 / 0.7900
FSRCNN	3	33.16 / 0.9140	29.43 / 0.8242	28.52 / 0.7893	26.42 / 0.8064
MFSR	3	**33.76 / 0.9217**	**29.82 / 0.8318**	**28.82 / 0.7976**	**27.06 / 0.8266**
Bicubic	4	28.42 / 0.8104	26.00 / 0.7027	25.96 / 0.6675	23.14 / 0.6577
A+	4	30.28 / 0.8603	27.32 / 0.7491	26.82 / 0.7087	24.32 / 0.7183
SRCNN	4	30.48 / 0.8628	27.49 / 0.7503	26.90 / 0.7101	24.52 / 0.7221
SelfExSR	4	30.31 / 0.8619	27.40 / 0.7518	26.84 / 0.7106	24.79 / 0.7374
RFL	4	30.14 / 0.8548	27.24 / 0.7451	26.75 / 0.7054	24.19 / 0.7096
FSRCNN	4	30.71 / 0.8657	27.59 / 0.7535	26.96 / 0.7128	24.60 / 0.7258
MFSR	4	**31.49 / 0.8841**	**28.08 / 0.7686**	**27.29 / 0.7254**	**25.11 / 0.7513**

Our model is tested on four benchmark data sets, which are Set5 [9], Set14 [10], BSD100 [11] and Urban100 [12]. The ground truth images are downscaled by bicubic interpolation to generate LR/HR image pairs for both training and testing databases. We convert each color image into the YCbCr color space and only process the luminance channel with our model, and bicubic interpolation is applied to the chrominance channels, because the visual system of human is more sensitive to details in intensity than in color.

4.2 Comparisons with the State-of-the-arts

We compare the proposed MFSR with 5 state-of-the-art SR algorithms: A+ [13], SRCNN [1], SelfExSR [12], REL [14] and FSRCNN [4]. Table 1 shows the average PSNR and SSIM [15] of the results by different SR methods. It can be observed that the proposed method achieves the best SR performance in all experiments.

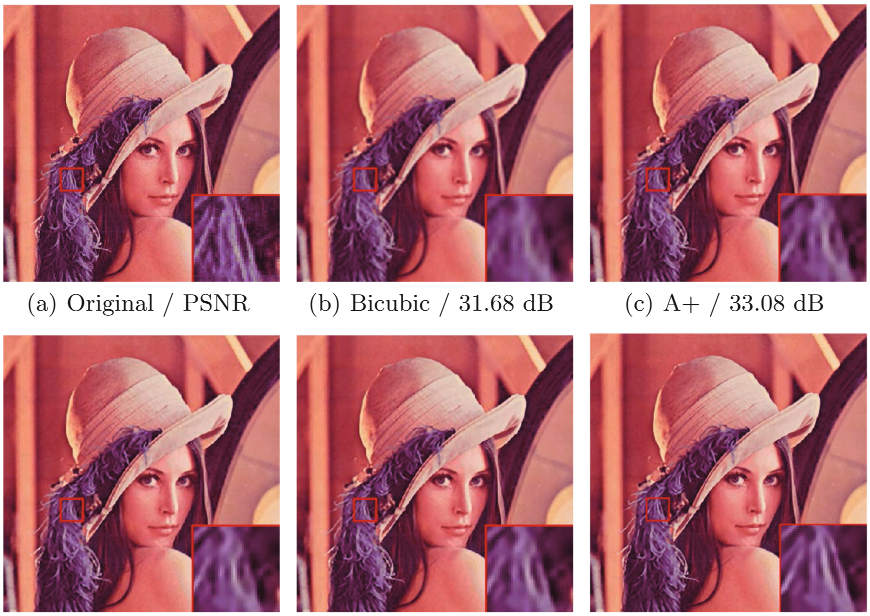
Figure 2 shows visual comparisons on image "butterfly" with a scale factor of 3×. Our method accurately reconstructs the very thin line on the butterfly. We observe that methods using the bicubic upsampling for pre-processing generate results with noticeable artifacts. In contrast, our approach effectively suppresses such artifacts. Similarly, in Figs. 3, 6 and 7, contours are clean in our method whereas they are severely blurred or distorted in other methods.



(a) Original / PSNR (b) Bicubic / 24.04 dB (c) A+ / 25.90 dB

(d) SRCNN-Ex / 27.95 dB (e) FSRCNN / 28.68 dB (f) MFSR / **29.98 dB**

Fig. 2. The "butterfly" image from the Set5 dataset with an upscaling factor 3.



(a) Original / PSNR (b) Bicubic / 31.68 dB (c) A+ / 33.08 dB

(d) SRCNN-Ex / 33.67 dB (e) FSRCNN / 33.85 dB (f) MFSR / **34.02 dB**

Fig. 3. The “lenna” image from the Set14 dataset with an upscaling factor 3.

4.3 Execution Time

Our network is implemented using the Python interface of *Caffe* package [6] and is trained and tested on a machine with 4.2 GHz Intel i7 CPU (32G RAM) and Nvidia TITAN X (Pascal) GPU (12G Memory). Table 2 shows the execution time of our model on Set5, Set14, BSD100 and Urban100 by using CPU implementation and GPU implementation. The speed of the proposed MFSR using GPU is much faster than that of using CPU. Figure 4 shows the trade-offs between the

Table 2. Comparison of the execution time (sec) of our method on the 4 benchmark datasets with CPU implementation and GPU implementation.

CPU / GPU	Scale	Set5	Set14	BSD100	Urban100
CPU	2	0.832	1.589	1.004	5.267
GPU	2	0.015	0.018	0.010	0.041
CPU	3	0.254	0.523	0.341	1.746
GPU	3	0.012	0.013	0.007	0.023
CPU	4	0.147	0.301	0.187	0.972
GPU	4	0.009	0.010	0.005	0.016

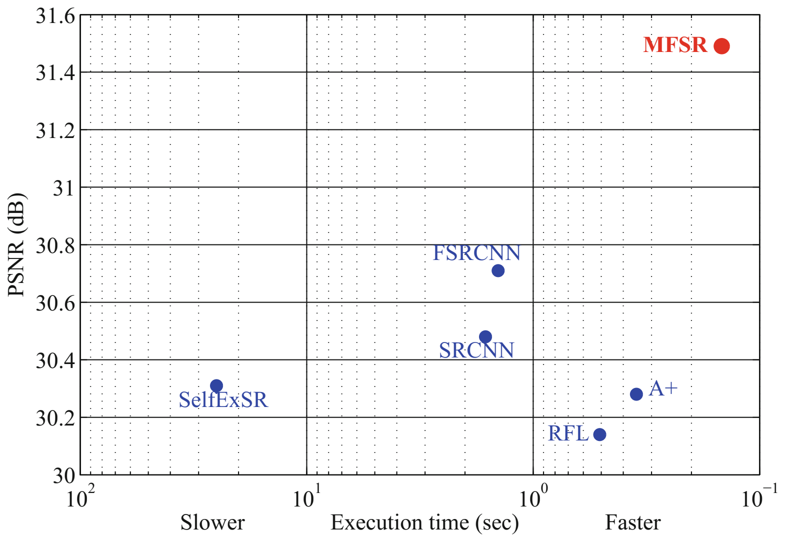


Fig. 4. The average PSNR and the average inference time for upscaling factor 4× on Set5. SRCNN and FSRCNN use the public slower MATLAB implementation of CPU. Our MFSR uses the matcaffe implementation of CPU.

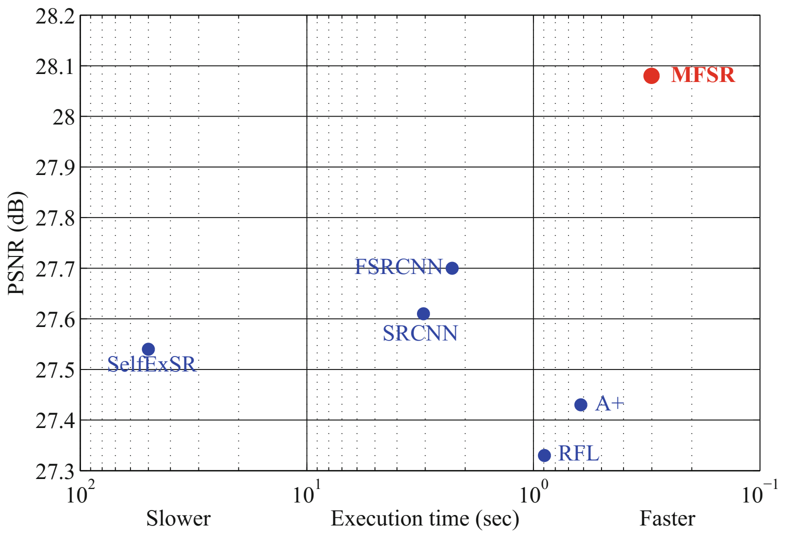
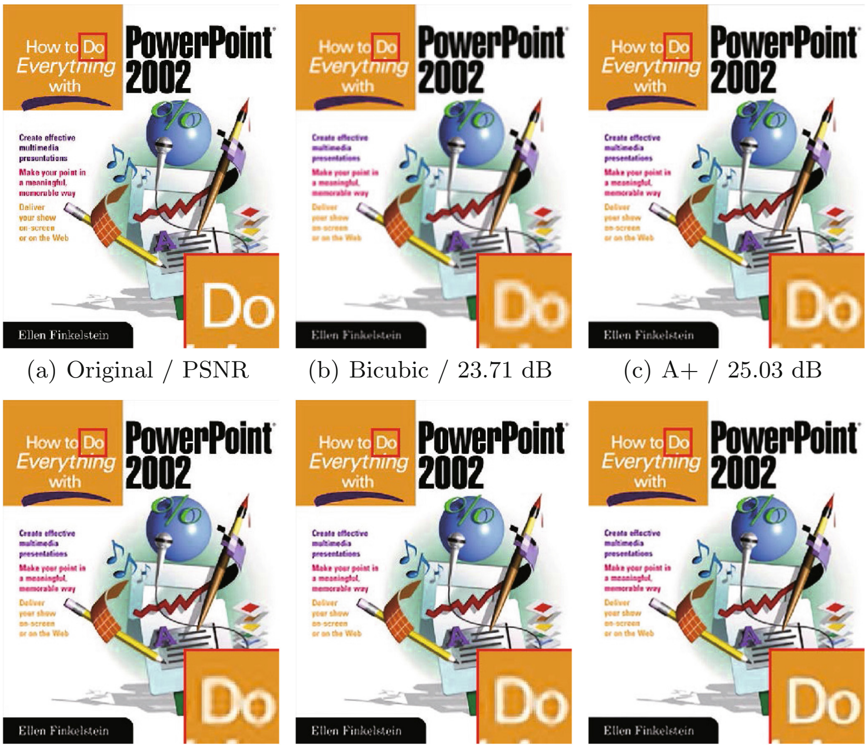


Fig. 5. The average PSNR and the average inference time for upscaling factor 4× on Set14. All methods use CPU implementation.

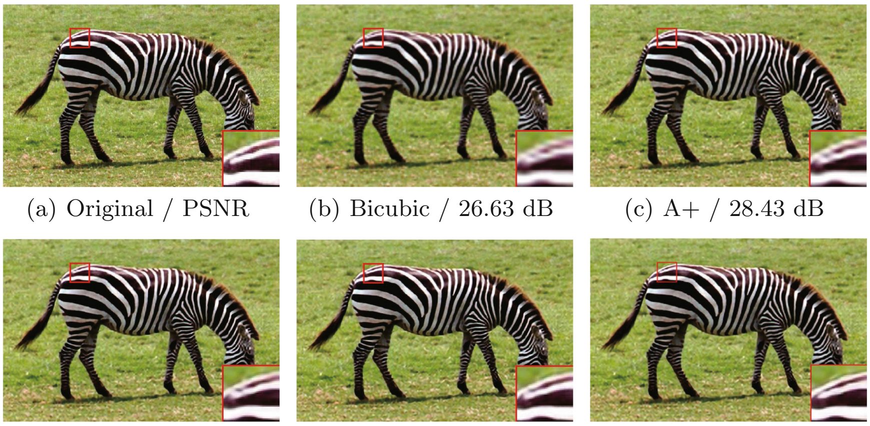
run time and performance on Set5 for $4 \times$ SR. We measure the run time of all methods on CPU. Similarly, from Fig. 5, we can see that the proposed MFSR generates SR images efficiently and accurately on Set14 with scaling factor 4.



(a) Original / PSNR (b) Bicubic / 23.71 dB (c) A+ / 25.03 dB

(d) SRCNN-Ex / 27.04 dB (e) FSRCNN / 27.11 dB (f) MFSR / **28.05 dB**

Fig. 6. The “ppt3” image from the Set14 dataset with an upscaling factor 3.



(a) Original / PSNR (b) Bicubic / 26.63 dB (c) A+ / 28.43 dB

(d) SRCNN-Ex / 29.29 dB (e) FSRCNN / 29.42 dB (f) MFSR / **29.83 dB**

Fig. 7. The “zebra” image from the Set14 dataset with an upscaling factor 3.

5 Conclusions

In this work, we investigate a deep convolutional network for single image super-resolution. Our method presents the multi-context fusion leaning and combine it with residual learning. MFSR is compared with several state-of-art SR methods (both DL and non-DL) in our experiments, and shows a visible performance advantage both quantitatively and perceptually. In the future, this approach of image super-resolution will be explored to facilitate other image restoration problems such as denoising and compression artifacts reduction.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grant 61472304, 61432014 and U1605252.

References

1. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. TPAMI **38**(2), 295–307 (2015)
2. Wang, Z., Liu, D., Yang, J., Han, W., Huang, T.: Deep networks for image super-resolution with sparse prior. In: ICCV, pp. 370–378 (2015)
3. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks. In: CVPR, pp. 1646–1654 (2016)
4. Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 391–407. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_25
5. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV (2015)
6. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: ACM MM, pp. 675–678 (2014)
7. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. TIP **19**(11), 2861–2873 (2010)
8. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV (2001)
9. Bevilacqua, M., Roumy, A., Guillemot, C., AlberiMorel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: BMVC (2012)
10. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Boissonnat, J.-D., Chenin, P., Cohen, A., Gout, C., Lyche, T., Mazure, M.-L., Schumaker, L. (eds.) Curves and Surfaces 2010. LNCS, vol. 6920, pp. 711–730. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27413-8_47
11. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. TPAMI **33**(5), 898–916 (2011)
12. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: CVPR, pp. 5197–5206 (2015)

13. Timofte, R., De Smet, V., Van Gool, L.: A+: adjusted anchored neighborhood regression for fast super-resolution. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9006, pp. 111–126. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16817-3_8
14. Schulter, S., Leistner, C., Bischof, H.: Fast and accurate image upscaling with super-resolution forests. In: CVPR, pp. 3791–3799 (2015)
15. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. TIP **13**(4), 600–612 (2004)