# Efficient Single Image Super-Resolution via Hybrid Residual Feature Learning with Compact Back-Projection Network

Feiyang Zhu, Qijun Zhao
College of Computer Science, Sichuan University, Chengdu, Sichuan, P. R. China
feiyangzhu@stu.scu.edu.cn; qjzhao@scu.edu.cn

## Abstract

*Deep learning methods have achieved state-of-the-art accuracy in single image super-resolution (SISR). Yet, how to achieve good balance between efficiency and accuracy in SISR is still an open issue. While most existing methods learn residual features only in low resolution (LR) space in order for higher efficiency, recent studies show that jointly learning residual features in LR and high resolution (HR) space is more preferred for accurate SISR. In this paper, we propose an efficient SISR method via learning hybrid residual features, based on which the residual HR image can be reconstructed. To fulfill hybrid residual feature learning, we propose a compact back-projection network that can simultaneously generate features in both LR and HR space by cascading up- and down- sampling layers with small-sized filters. Extensive experiments on four benchmark databases demonstrate that our proposed method can achieve high efficiency (i.e., small number of parameters and operations) while preserving state-of-the-art SR accuracy.*

## 1. Introduction

As a low-level computer vision task, single image super-resolution (SISR) aims to restore a high-resolution (HR) image from a low-resolution (LR) image. SISR is an ill-posed problem because a lot of information is lost in the LR image compared with its HR counterpart. To solve this problem, a number of methods have been proposed, including non-learning based [6, 20] and learning based [5, 4, 12, 14, 7, 24, 9, 1]. Learning based methods generally work better by learning prior knowledge of the relation between HR and LR images from training data of pairing HR and LR images. Among them, deep learning based methods [9, 1, 7, 4] achieve the state-of-the-art.

Dong et al. [5] firstly introduced deep learning to SISR with a three-layer convolutional neural network. Since then, many efforts have been made to improve the SISR accuracy by exploring different network structures, e.g., residual networks in [13], deep back-projection networks (DBPN) in [7], and memory networks in [18]. Despite the impressive SISR accuracy of these methods, they usually have a large number of parameters and operations, which prevent them from being adopted in many real-world applications, particularly image super-resolution (SR) on mobile phones.


Figure 1: $\times 2$ Super-resolution results of our method (CBPN) and other algorithms. PSNR: CARN [1] (28.86 dB), FALSR-A [4] (28.80 dB), and Ours (29.37 dB).

In order to implement SISR on low-power systems (e.g., mobile phones and drones), the authors of DBPN [7] reduce the number of filters as well as the depth of the network in DBPN (resulting in light-weight DBPN, or in short DBPN-M/S) at the cost of degraded SR accuracy. Ahn et al. [1] propose to cascade residual networks with multiple shortcut connections for learning residuals in LR feature space (the corresponding SR model is called CARN), and further reduce the model complexity for mobile applications by using the recursive network scheme (the model is thus called CARN-M). Hui et al. [9] propose an information distillation network that can learn HR residual features in the LR feature space with lightweight parameters and low computational complexity. Instead of manually designing efficient networks, Chu et al. [4] employ neural architecture search to construct three lightweight SISR networks (i.e., FALSR-A, FALSR-B, FALSR-C). The building blocks of these networks are similar to the residual blocks in CARN, and therefore they essentially learn residuals in LR feature space too.

In this paper, we aim to propose an efficient SISR method by exploiting latest advancement in network structure design for SR. Specifically, according to DBPN [7], learning residuals in both LR and HR feature spaces is beneficial to obtaining more accurate SR results. However, for one thing, DBPN has high complexity because of its large-sized filters and elaborated network structure; for another, as discussed in the last paragraph, in order to improve the efficiency, existing methods all learn residuals only in LR feature space. Besides, according to [24] and [1], fully utilizing multi-level representations and multiple local features can effectively boost SR accuracy; and according to [9], reconstructing the HR image for a LR image by learning its residual in HR image space is an efficient way. However, DBPN reconstructs the HR image directly from the multiple HR features obtained via back-projection networks.

Motivated by the above observations, we propose a compact back-projection network that can learn hybrid residual features in both LR and HR spaces to achieve a good balance between efficiency and accuracy of SISR. We will introduce in detail our proposed method in Section 3 after briefly reviewing related work in Section 2. We then report in Section 4 our experimental results, which prove that our proposed method has high efficiency (i.e., fewer parameters and operations) while maintaining state-of-the-art SR accuracy. We finally conclude the paper in Section 5.

## 2. Related Work

Recently, deep learning methods have substantially push forward the frontier of image super-resolution. Dong et al. [5] first proposed a three convolution layer network, SRCNN, which maps the interpolated LR images to HR counterparts. Followup works focused on using deeper network, new learning strategies or network structures. VDSR [10] adopted a deeper network with global residual learning. DRRN [17] compressed the network depth by applying recursive learning strategy. Ledig et al. [13] employed generative adversarial network (GAN) for SISR and a perceptual loss to train SR networks. Tong et al. [21] used dense connections to transfer information between middle layers, which enabled features reuse in networks. Haris et al. [7] exploited multiple up- and down- sampling layers, which are alternately applied to implement bidirectional projection between LR and HR spaces. Lim et al. [14] connected multiple residual blocks to fulfill multi-scale deep SR with more than 160 layers and $8M$ parameters. Tai et al. [18] proposed a very deep persistent memory network (MemNet) for SISR, which mines persistent memory through an adaptive learning process. The primary goal of those SR methods is to achieve better SR accuracy. Yet, the large number of operations and parameters hinders their application in low power scenarios.

Ahn et al. [1] proposed a cascaded residual SR network (CARN). They compressed the model and reduced its computational complexity by effectively cascading residual blocks. Further they employ group convolution and pointwise convolution to construct a smaller model (CARN-M). Chu et al. [4] constructed three lightweight networks via automatic neural architecture search. In this work, we compress the SR network by reducing the size of feature channels, and reduce the number of operation by replacing the deconvolution layers with sub-pixel convolution layers [16]. In addition, the utilization of LR and HR features to rehabilitate the HR images guarantees the SR accuracy of our network. Finally, we establish the fast and lightweight networks, CBPN and CBPN-S, achieving comparable SR results with state-of-the-art methods.

## 3. Proposed Method

### 3.1. Network architecture

Figure 2 depicts the network structure of our proposed method (for $4\times$ super-resolution), namely compact back-projection network, or shortly CBPN. CBPN is mainly composed of three parts: Low-level feature extraction module, compact projection module and reconstruction module. Let $I_{LR}$ be the input LR image of $M^l \times N^l$ and $I_{HR}$ be the output HR image of $M^h \times N^h$, where $M^l < M^h$ and $N^l < N^h$. $F_L^0$ is the LR feature output by the low-level feature extraction module. $F_L^t$ and $F_H^t$, respectively, denote the LR feature and HR feature output by $t^{\text{th}}$ UD block (see Fig. 3). $F_L$ is obtained by compressing the LR features generated by all UD blocks. $F_H^0$ is obtained by upsampling $F_L$ with the first upsampling layer. $F_H$ is the output of the second upsampling layer. $\Delta I_H$ represents the HR residual image learned by our network. $I_H^0$ is the HR image obtained by upsampling the input $I_{LR}$ with bicubic interpolation. We train our network in an end-to-end manner.

In low-level feature extraction module, we use two convolution layers to map the input LR image to a feature vector $F_L^0$, which is the input to the following projection module. Several UD blocks, two compression layers (CL) and two upsampling layers constitute the compact projection module. We use dense connections among the UD blocks and export the LR and HR features of UD blocks for subsequent processing. The operation of the first compression layer can be represented by $F_L = f'_{compress}([F_L^1, \ldots, F_L^t, \ldots, F_L^T])$, which concatenates all input LR features and feeds the concatenated features to a $1 \times 1$ convolution layer. The operation of the first upsampling layer is $F_H^0 = f'_{up}(F_L)$, which defines the mapping from LR space to HR space and whose input is the compressed LR feature. $F_H = f''_{up}(f''_{compress}[F_H^0, F_H^1, F_H^t, F_H^T])$, where $f''_{compress}$ is the compression layer between the two upsampling layers, and $f''_{up}$ is the second upsampling layer. In our $4\times$ network,

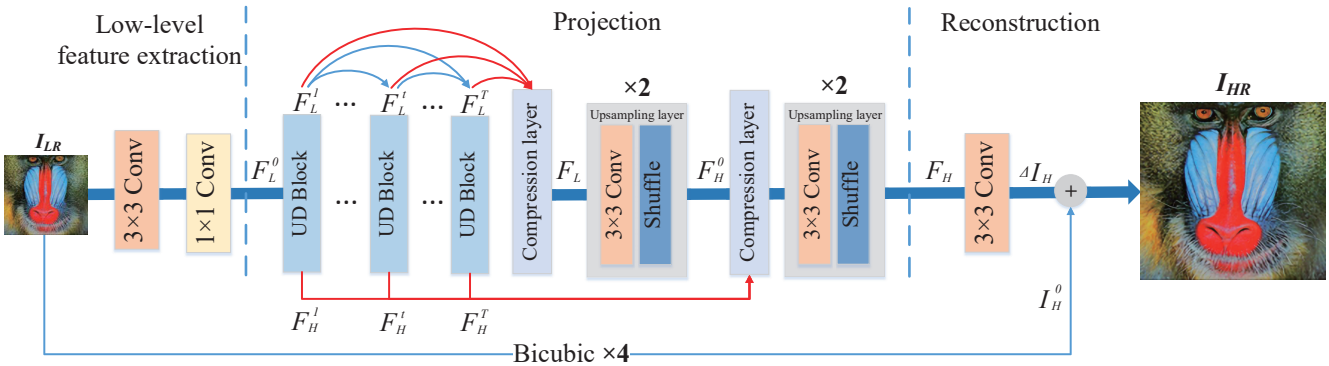


Figure 2: Network structure of our CBPN on $\times 4$ scale. CBPN consists of three modules: low-level feature extraction, projection and reconstruction. Blue arrow lines in projection module are dense connection between UD-blocks. Red arrow lines are the output of UD blocks.

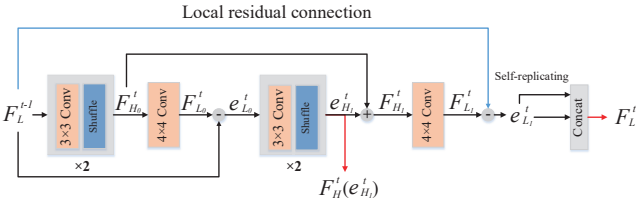


Figure 3: The detailed structure of UD block.

$T = 8$ and $t = 4$, and thus only part of HR features are involved in reconstruction. In reconstruction module, we employ a $3 \times 3$ convolution layer as reconstruction layer. The output of reconstruction layer, $\Delta I_H = f_{rec}(F_H)$, is a residual image $\Delta I_H$ of high resolution. The final output HR image of our network is $I_{HR} = \Delta I_H + I_H^0$.

### 3.2. Projection module

Our compact projection module is inspired by the up- and down- projection units of DBPN [7]. But we adapt the projection units of DBPN from the following three aspects to improve its efficiency without significant loss of SR accuracy. (i) The LR features generated by down-projection units are overlooked during reconstruction in DBPN, while we utilize both HR and LR features. (ii) DBPN implements upsampling layers with deconvolutional layers, while we employ sub-pixel convolutional layers, which have lower complexity. (iii) We add local residual connections to improve the learning ability of the UD blocks.

Figure 3 shows the detailed structure of our UD block. $F_L^{t-1}$ represents the input of the $t^{\text{th}}$ UD block. $F_L^t$ and $F_H^t$, respectively, denote the LR and HR feature outputs of $t^{\text{th}}$ UD block. $F_{L_0}^t$ and $F_{L_1}^t$ are the LR features. $F_{H_0}^t$ and $F_{H_1}^t$ are the HR features. $e_{L_0}^t$ and $e_{L_1}^t$ are LR residual features, and $e_{H_1}^t$ is HR residual feature. The operations involved in our projection unit can be defined as follows

$$F_{H_0}^t = U_0^t(F_L^{t-1}; W_U^0), \tag{1}$$

where $U_0^t$ denotes the first sub-pixel convolution layer in $t^{\text{th}}$ UD block, and $W_U^0$ is the weights of the first sub-pixel convolution layer.

$$F_{L_0}^t = C_0^t(F_{H_0}^t; W_C^0), \tag{2}$$

where $C_0^t$ denotes $4 \times 4$ convolution operation with weights $W_C^0$. This convolution layer implements the mapping from HR space to LR space.

$$e_{L_0}^t = F_{L_0}^t - F_L^{t-1}, \tag{3}$$

$$F_H^t = e_{H_1}^t = U_1^t(e_{L_0}^t; W_U^1), \tag{4}$$

where $U_1^t$ is the second upsampling layer in $t^{\text{th}}$ UD block, and $W_U^1$ represents its weights. The LR residual feature $e_{L_0}^t$ is mapped to HR space by this upsampling layer. $e_{H_1}^t$ is the HR residual feature, which is also taken as the HR feature output $F_H^t$ of this UD block.

$$F_{H_1}^t = e_{H_1}^t + F_{H_0}^t, \tag{5}$$

$$F_{L_1}^t = C_1^t(F_{H_1}^t; W_C^1), \tag{6}$$

where $C_1^t$ is the operation of $4 \times 4$ convolution layer, and $C_1^t$ is similar to $C_0^t$, mapping features from HR to LR space.

$$e_{L_1}^t = F_{L_1}^t - F_L^{t-1}, \tag{7}$$

$$F_L^t = Concat([e_{L_1}^t, e_{L_1}^t]). \tag{8}$$

We duplicate the residual feature ($e_{L_1}^t$) and stack it with its duplication together as LR residual feature output ($F_L^t$) of the UD block. The duplication operation aims to increase the weight of LR features in the network.

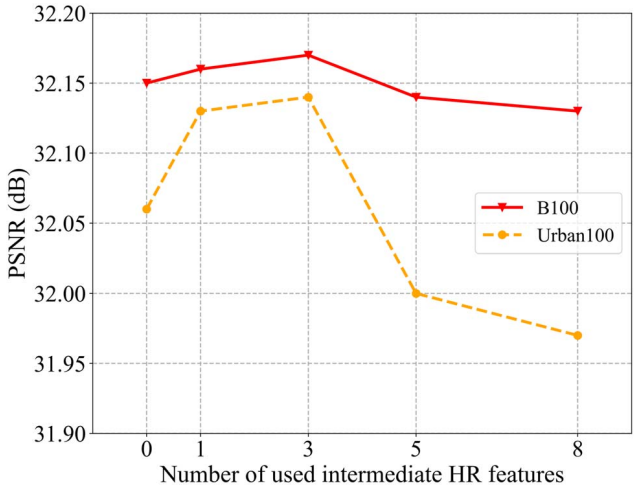


Figure 4: SR accuracy in terms of PSNR of our CBPN for $\times 2$ SR on B100 and Urban100 datasets w.r.t. the number of used intermediate HR features generated by the UD blocks.

The whole process of UD blocks embodies the idea of back-projection. The cascaded upsampling and downsampling layers achieve iteratively learning residual features. The information in both HR and LR space is expanded through those projection units. The iterative operation can be interpreted as a self-correcting procedure which provides feedback of the projection error to sampling layer and iteratively improves the features, as mentioned in [7].

### 3.3. Reconstruction module

Reconstruction module combines the features in LR and HR space such that a wealth of feature information can be used to recover the HR image. Note that the HR features generated in both low and high level layers of the UD blocks are used by the reconstruction module. This is beneficial to reconstructing high quality HR images. In addition, the features from LR space provide additional low frequency information, promoting effective reconstruction together with HR features.

## 4. Experimental Results

### 4.1. Datasets and metrics

We train our proposed network with a high-quality dataset DIV2K [19] of 800 training images, 100 validation images and 100 test images, and test the obtained model on four standard benchmark datasets: Set5 [2], Set14 [23], BSDS100 [15] and Urban100 [8]. Set5, Set14 and BSDS100 mainly consist of images of natural scenes, while Urban100 consists of images of urban scenes. LR images are obtained by bicubic downsampling the HR images. We use PSNR and SSIM [22] to measure the SR accuracy,

| | CBPN-L | CBPN-H | CBPN |
|---|---|---|---|
| HR features | | √ | √ |
| LR features | √ | | √ |
| Set5 | 37.87 | 37.86 | **37.90** |
| B100 | 32.15 | 32.13 | **32.17** |
| Urban100 | 32.06 | 32.10 | **32.14** |

Table 1: SR accuracy in terms of PSNR (dB) of our CBPN with or without using LR/HR features on three benchmark datasets for $\times 2$ SR. CBPN-L means using only LR features, and CBPN-H using only HR features. CBPN uses both LR and HR features.

| Algorithm | Mult-Adds | Set5 PSNR/SSIM | Set14 PSNR/SSIM |
|---|---|---|---|
| D-DBPN-L [7] | 1101.7G | 31.99/0.893 | 28.52/0.778 |
| CBPN | 97.9G | 32.21/0.894 | 28.63/0.781 |

Table 2: Quantitative comparison results between our CBPN and D-DBPN-L [7] for $\times 4$ SR. The best results are shown in red color.

and the number of parameters in the SR models and the total number of multiplications and summations required to super-resolution one image (denoted as Mult-Adds) to measure the computational complexity of the SR models. Note that we calculate PSNR and SSIM on the luminance channel (Y) of the YCbCr color space.

### 4.2. Implementation details

We use $L1$ loss instead of $L2$ loss to train our model. We initialize the learning rate to $10^{-4}$ for all layers and decrease it by a factor of 10 for every $1M$ iterations. We augment the training images by randomly flipping them horizontally or vertically and rotating them by $90°$. In our experiments, the input LR image size is $64 \times 64$, and the batch size is set to eight. Note that CBPN is implemented with eight UD blocks with dense connections, while CBPN-S (i.e., a simplified version of CBPN) has only three UD blocks. The training is done on a PC with a NVIDIA RTX 2080TI GPU, and stopped at $2M$ iterations. During the training, we export a model every 50 epochs. We finally choose the model that has the highest PSNR on the validation set.

### 4.3. Model analysis

We first study the impact of the number of used intermediate HR features in the UD blocks on the SR accuracy. We construct five networks, which use $0, 1, 3, 5$, and $8$ HR features, respectively. We evaluate their $\times 2$ SR accuracy (in terms of PSNR) on the B100 and Urban100 datasets. The results are shown in Fig. 4. As can be seen, although

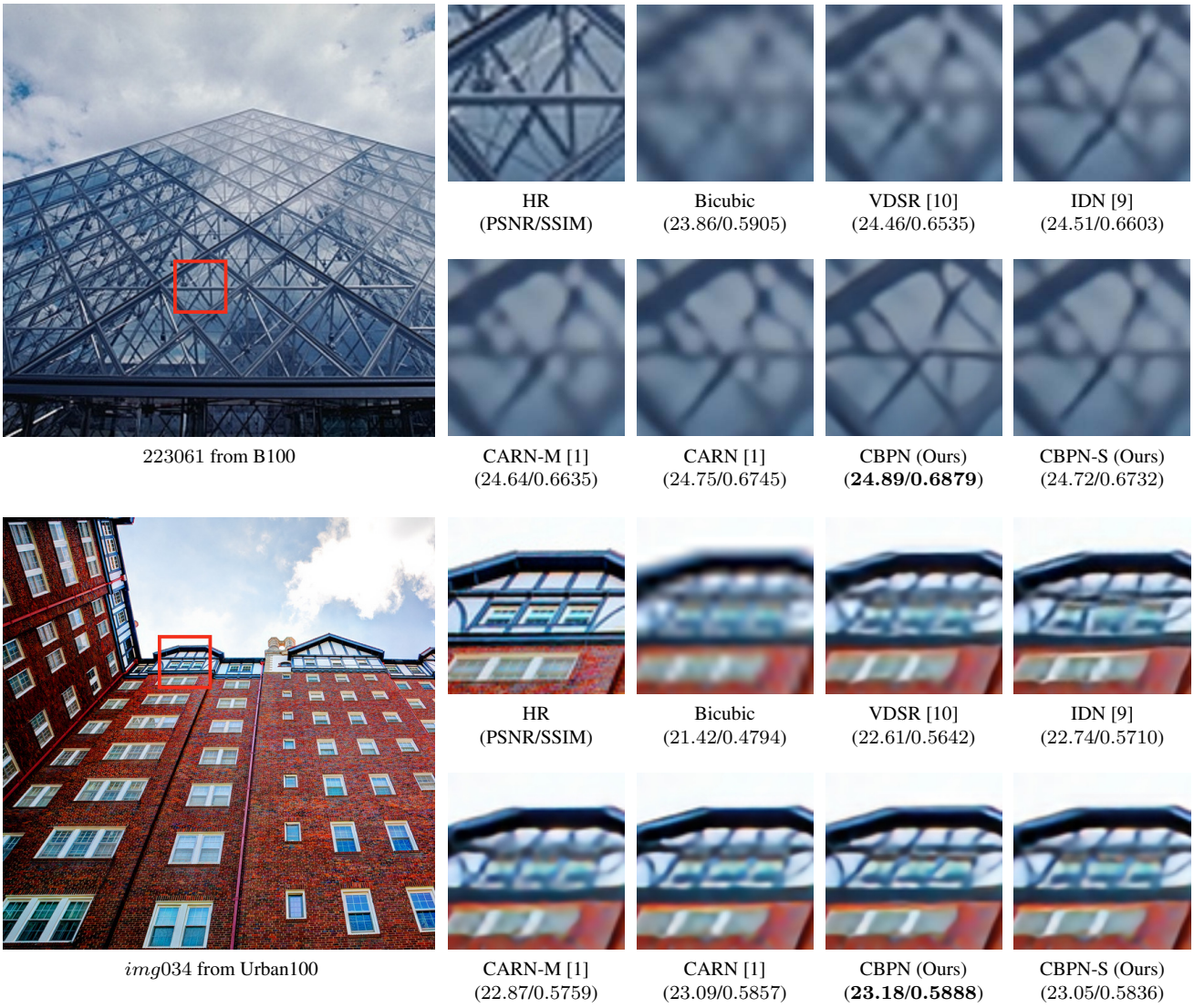


Figure 5: Comparison of our CBPN with other SR methods for $\times 4$ SR. Bold text represents best performance in terms of PSNR and SSIM.

HR features do help for obtaining better HR images, using more HR features does not necessarily mean higher SR accuracy. This indicates that a balance should be made in utilizing LR and HR features. According to the results in Fig. 4, when three HR features (i.e., $F_H^1, F_H^4, F_H^8$ in CBPN, and all the HR features in CBPN-S) are used, the best SR accuracy is achieved on both datasets. Therefore, in the rest experiments, when HR features are used, we use three HR features as being done here.

We further evaluate the contribution of HR and LR features on three datasets, Set5, B100 and Urban100. The results are given in Table 1, in which CBPN-L means using only LR features, CBPN-H using only HR features, and CBPN using both LR and HR features. Obviously, the best SR accuracy is achieved when both LR and HR features are used. This suggests that HR and LR feature spaces contain complementary information that are useful for image SR.

Besides, we also analyze the complexity of our proposed CBPN. Table 2 compares both complexity and accuracy of CBPN and one of the state-of-the-art (SOTA) models, D-DBPN-L [7], for $\times 4$ SR on the Set5 and Set14 datasets. In this experiment, following [1], the HR images are assumed to be $720p$ ($1280 \times 720$). Under this setting, the Multi-Adds of CBPN is $97.9G$, which is about $9\%$ of that of D-DBPN-L ($1101.7G$). As for the number of parameters involved in the models, our CBPN has $1,197K$ parameters, while D-DBPN-L has about two times more parameters (i.e., $2,198K$). Moreover, our CBPN achieves better

| Scale | Model | Params | Mult-Adds | Set5 PSNR/SSIM | Set14 PSNR/SSIM | B100 PSNR/SSIM | Urban100 PSNR/SSIM |
|---|---|---|---|---|---|---|---|
| ×2 | SRCNN [5] | 57K | 52.7G | 36.66/0.9542 | 32.42/0.9063 | 31.36/0.8879 | 29.50/0.8946 |
| | VDSR [10] | 665K | 612.6G | 37.53/0.9587 | 33.03/0.9124 | 31.90/0.8960 | 30.76/0.9140 |
| | LapSRN [12] | 813K | 29.9G | 37.52/0.9590 | 33.08/0.9130 | 31.80/0.8950 | 30.41/0.9100 |
| | DRRN [17] | 297K | 6,796.9G | 37.74/0.9591 | 33.23/0.9136 | 32.05/0.8973 | 31.23/0.9188 |
| | SelNet [3] | 974K | 225.7G | 37.89/0.9598 | 33.61/0.9160 | 32.08/0.8984 | - |
| | IDN [9] | 553K | 202.8G | 37.83/0.9600 | 33.30/0.9148 | 32.08/0.8985 | 31.27/0.9196 |
| | CARN [1] | 1,592K | 222.84G | 37.76/0.9590 | 33.52/0.9166 | 32.09/0.8978 | 31.92/0.9256 |
| | CARN-M [1] | 412K | 91.2G | 37.53/0.9583 | 33.26/0.9141 | 31.92/0.8960 | 30.83/0.9233 |
| | FALSR-A [4] | 1,021K | 234.7G | 37.82/0.9595 | 33.55/0.9168 | 32.12/0.8987 | 31.93/0.9256 |
| | FALSR-B [4] | 326K | 74.7G | 37.61/0.9585 | 33.29/0.9143 | 31.97/0.8967 | 31.28/0.9191 |
| | FALSR-C [4] | 408K | 93.7G | 37.66/0.9586 | 33.26/0.9140 | 31.96/0.8965 | 31.24/0.9187 |
| | CBPN (Ours) | 1,036K | 240.7G | 37.90/0.9590 | 33.60/0.9171 | 32.17/0.8989 | 32.14/0.9279 |
| | CBPN-S (Ours) | 430K | 101.5G | 37.69/0.9583 | 33.36/0.9147 | 32.02/0.8972 | 31.55/0.9217 |
| ×4 | SRCNN [5] | 57K | 52.7G | 30.48/0.8628 | 27.49/0.7503 | 26.90/0.7101 | 24.52/0.7221 |
| | VDSR [10] | 665K | 612.6G | 31.35/0.8838 | 28.01/0.7674 | 27.29/0.7251 | 25.18/0.7524 |
| | DRCN [11] | 1,774K | 9,788.7G | 31.53/0.8854 | 28.02/0.7670 | 27.23/0.7233 | 25.14/0.7510 |
| | LapSRN [12] | 813K | 149.4G | 31.54/0.8850 | 28.19/0.7720 | 27.32/0.7280 | 25.21/0.7560 |
| | DRRN [17] | 297K | 6,796.9G | 31.68/0.8888 | 28.21/0.7720 | 27.38/0.7284 | 25.44/0.7638 |
| | SelNet [3] | 1,417K | 83.1G | 32.00/0.8931 | 28.49/0.7783 | 27.44/0.7325 | - |
| | IDN [9] | 553K | 89.0G | 31.82/0.8903 | 28.25/0.7730 | 27.41/0.7297 | 25.41/0.7632 |
| | SRDenseNet [21] | 2,015K | 389.9G | 32.02/0.8934 | 28.50/0.7782 | 27.53/0.7337 | 26.05/0.7819 |
| | CARN [1] | 1,592K | 90.9G | 32.13/0.8937 | 28.60/0.7806 | 27.58/0.7349 | 26.07/0.7837 |
| | CARN-M [1] | 412K | 32.5G | 31.92/0.8903 | 28.42/0.7762 | 27.44/0.7304 | 25.63/0.7688 |
| | CBPN (Ours) | 1,197K | 97.9G | 32.21/0.8944 | 28.63/0.7813 | 27.58/0.7356 | 26.14/0.7869 |
| | CBPN-S (Ours) | 592K | 63.1G | 31.93/0.8908 | 28.50/0.7785 | 27.50/0.7324 | 25.85/0.7772 |

Table 3: Quantitative results of our methods compared with the SOTA SR methods. Red and second texts indicate the best and the second best results.

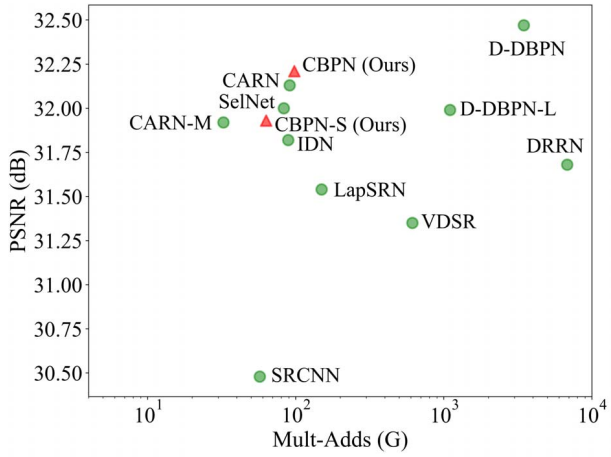


Figure 6: Trade-off between SR accuracy (PSNR) and model complexity (the number of operations or Mult-Adds) for ×4 SR on the Set5 dataset. Red triangles represent our methods and green circles are other methods.

SR accuracy in terms of both PSNR and SSIM. These results prove that our method makes a better balance between efficiency and accuracy of SR than the counterpart method.

### 4.4. Comparison with the SOTA methods

We quantitatively and qualitatively compare our CBPN and CBPN-S with SOTA methods. Table 3 reports the PSNR and SSIM of different methods for $times2$ and ×4 SR on the four test datasets. Our proposed CBPN achieves the highest accuracy compared with the SOTA methods for ×4 SR on all the four datasets. For ×2 SR, our methods also perform competitively. CBPN achieves 32.14 dB PSNR on Urban100, 0.21 dB better than FALSR-A [4] and 0.22 dB better than CARN [1]. On B100 CBPN is 0.05 dB better than FALSR-A [4] and 0.08 dB better than CARN [1]. Figs. 5 and 7 show the qualitative comparison results of CBPN, CBPN-S and other SOTA SR algorithms. Our methods tend to restore more realistic textures. Taking the image $img062$ in Fig. 7 for example, our CBPN successfully restores the correct building appearance, whereas the restored appearance of other methods are obviously different from

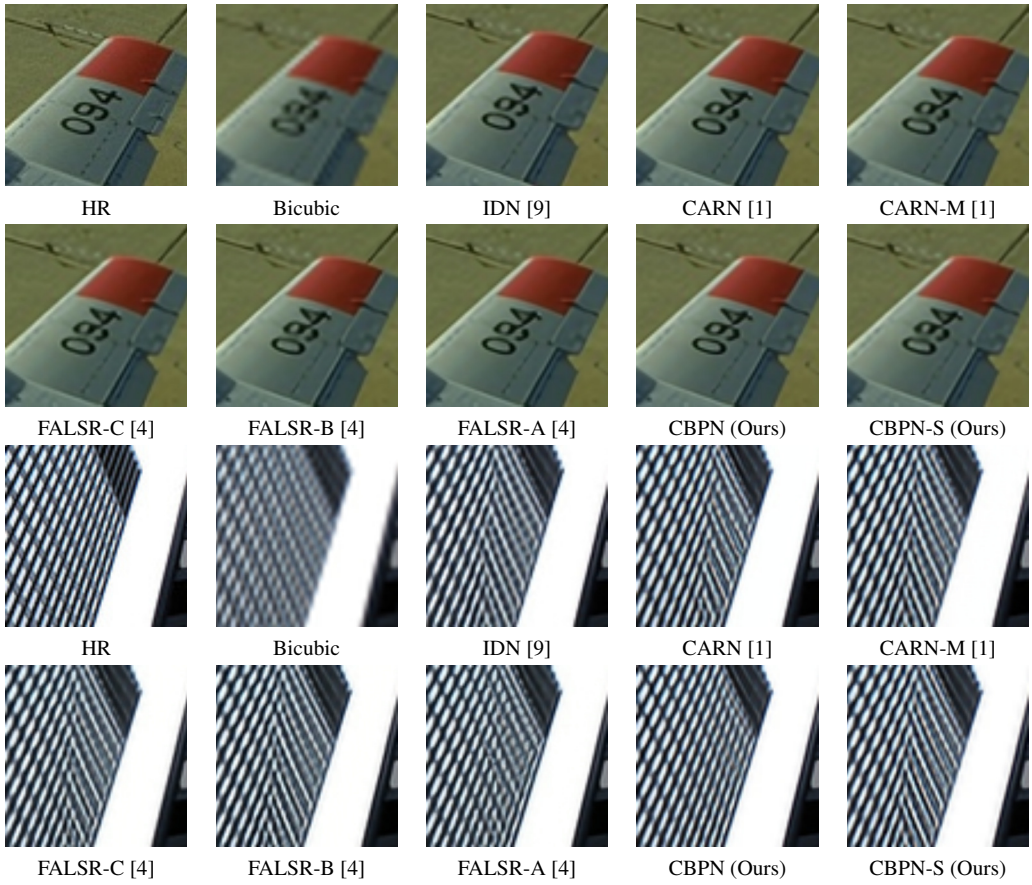


Figure 7: Visual comparison of the $\times 2$ super-resolution results of our CBPN and the SOTA algorithms for images $37073$ from B100 and $img062$ from Urban100.

the ground truth HR image.

Fig. 6 shows the comparison results with other deep learning based SR methods in terms of the number operations (Multi-Adds) required for $\times 4$ SR with respect to the SR accuracy in terms of PSNR on the Set5 dataset. As can be seen, Multi-Adds of our CBPN is similar to that of CARN and FALSR-A, but our CBPN achieves better SR accuracy than them. Note that the number of parameters of CBPN is $0.4M$ less than CARN [1]. Using fewer number of UD blocks and thus fewer parameters and operations, our simplified model CBPN-S obtains comparable accuracy with IDN [9] and D-DBPN-L [7] but much lower computational complexity. This demonstrates the better trade-off of our methods between complexity and accuracy.

## 5. Conclusion

In this paper, we propose a compact back-projection network for fast and effective single image super-resolution. We design novel UD blocks and reconstruction layer to restore the HR images by utilizing both HR and LR features. Our qualitative and quantitative evaluation results demonstrate that our method performs favorably against SOTA SR methods in terms of both accuracy and complexity. In the future, we will extend our method to other tasks, such as video SR that demands for restoring high-quality video in real time.

## References

[1] N. Ahn, B. Kang, and K. Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, pages 256–272. Springer, 2018.

[2] M. Bevilacqua, A. Roumy, C. Guillemot, and M. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, pages 1–10, 2012.

[3] J. Choi and M. Kim. A deep convolutional neural network with selection units for super-resolution. In *CVPRW*, pages 1150–1156, 2017.

[4] X. Chu, B. Zhang, H. Ma, R. Xu, J. Li, and Q. Li. Fast, accurate and lightweight super-resolution with neural architecture search. *arXiv preprint arXiv:1901.07261*, 2019.

[5] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 38(2):295–307, 2016.

[6] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.

[7] M. Haris, G. Shakhnarovich, and N. Ukita. Deep back-projection networks for super-resolution. In *CVPR*, pages 1664–1673, 2018.

[8] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015.

[9] Z. Hui, X. Wang, and X. Gao. Fast and accurate single image super-resolution via information distillation network. In *CVPR*, pages 723–731, 2018.

[10] J. Kim, J. K. Lee, and K. M. Lee. Accurate image superresolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016.

[11] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, pages 1637–1645, 2016.

[12] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, pages 5835–5843, 2017.

[13] 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. In *CVPR*, pages 105–114, 2017.

[14] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, pages 1132–1140, 2017.

[15] D. R. Martin, C. 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*, volume 2, pages 416–423, 2001.

[16] 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*, pages 1874–1883, 2016.

[17] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, pages 2790–2798, 2017.

[18] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, pages 4549–4557, 2017.

[19] R. Timofte, E. Agustsson, L. V. Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, pages 1110–1121, 2017.

[20] R. Timofte, V. De, and L. V. Gool. Anchored neighborhood regression for fast example-based super-resolution. In *ICCV*, pages 1920–1927, 2013.

[21] T. Tong, G. Li, X. Liu, and Q. Gao. Image super-resolution using dense skip connections. In *ICCV*, pages 4809–4817, 2017.

[22] 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.

[23] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730. Springer, 2010.

[24] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image super-resolution. In *CVPR*, pages 2472–2481, 2018.