LOW COMPLEXITY SINGLE IMAGE SUPER-RESOLUTION WITH CHANNEL SPLITTING AND FUSION NETWORK

Minqiang Zou, Jie Tang, Gangshan Wu

State Key Laboratory for Novel Software Technology, Nanjing University, China

ABSTRACT

Recently, deep convolutional neural networks (CNNs) have made remarkable progress on single image super-resolution (SISR). However, many of these methods use very deep or wide convolutional layers to achieve good performance, which treat all feature channels indiscriminately and neglect the difference among the contribution of each channel to the output results. In this paper, we propose a low complexity solution based on channel splitting and fusion network (CSFN) to address this problem. Our method uses channel splitting and channel fusion to enhance feature maps and make full use of valuable information, and then multiple residual channel splitting and fusion blocks (CSFB) are cascaded to continuously extract more important information for reconstruction. To further minimize redundant parameters and improve efficiency, we adopt group and recursive convolutional layer strategy in CSFB. Experiments demonstrate that our proposed CSFN could achieve higher performance with low computational complexity than most state-of-the-art methods.

Index Terms— Image super-resolution, convolutional neural networks, deep learning

1. INTRODUCTION

Single image super-resolution (SISR) is a classical computer vision task that generates a high-resolution (HR) image from a low-resolution (LR) one. Generally, SISR is regarded as an ill-posed problem since there are numerous HR images corresponding to one LR image. Several varieties of SR methods have been proposed to solve this inverse problem, such as interpolation-based methods, prediction-based methods [1, 2, 3], and example-based methods [4, 5, 6, 7]. In recent years, with the rapid development of computer performance and deep learning techniques, using deep learning to handle SR task is becoming a hot topic. These data-driven approaches have been proven to be able to achieve better results both in reconstruction accuracy and perceptual quality [8, 9].

Many previous works have proposed different efficient network structures for SR problem. Dong et al. [7] first propose a 3-layer CNN model named SRCNN, which has a lightweight structure and superior accuracy compared with the conventional example-based method. As stacking convolutional layers can increase the expressive ability of the network, Kim et al. [10] propose a very deep CNN model called VDSR. VDSR uses an extremely high learning rate and residual learning to ease the training difficulty in the deep model. FSRCNN [11] upsamples images using a deconvolution layer at the end of the network, which is much faster than the pre-upsampling method since it performs most of the mappings in low-dimensional space. ESPCN [12] uses sub-pixel convolution to learn the upscaling filters between LR and HR images, which is much faster than deconvolution. To encourage feature reuse and better propagate information, ResNet [13] and DenseNet [14] architecture are also widely used in SR tasks[15, 16, 17, 18], those methods can get superior performance by using deeper or wider networks.

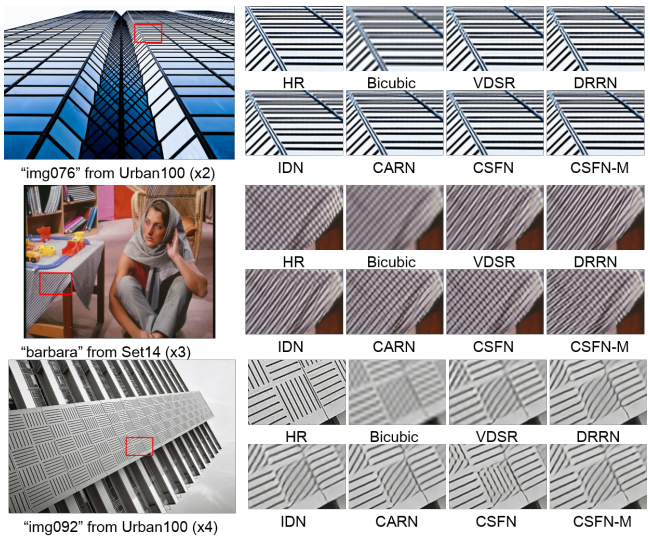

Fig. 1. Qualitative comparisons of our methods with other state-of-the-arts. our CSFN can reconstruct images more accurately and less blurring.

Although a deeper network can get better results, it is hard to apply this huge model to practical applications due to the heavy computational and memory costs. To control the number of network parameters, recursive convolutional layers are used in DRCN [19], DRRN [20]. Usually, a recursive network would increase the depth of the model. Thus excessive recursive operations that saving intermediate results would lead to running out of the GPU memory when processing

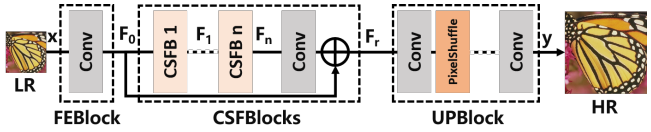

Fig. 2. Network architecture of the proposed CSFN. The dots in UPBlock represent multiple Conv-PixelShuffle module depend on scaling factor.

large images such as the case in DRRN[20], which is not suitable for real-world scenarios. CARN [21] is an accurate and lightweight deep network using the cascading mechanism to extract features. Chu et. al. [22] propose a neural architecture search technology to produce a lightweight model, but this search technology is also based on a specific network structure to reduce searching space.

However, most CNN-based methods treat all feature channels indiscriminately and neglect the different contributions of each channel to the output results. Hu et. al. [23] adopt a squeeze-and-excitation (SE) block to explicitly modeling the interdependencies between the channels of its convolutional features. RCAN [16] uses this channel attention mechanism in SR task to enhance discriminative learning ability, but all feature maps would flow into the next layer without distinction, which cannot make full use of useful channel features. To handle this inefficiency, we propose a new structure to consider the channel unevenness. In specific, we propose a deep residual channel splitting and fusion network (CSFN), which splits the channels to guide some features to generate more features in order to enhance feature maps gradually and make full use of the valuable information. In summary, our contributions can be summarized as follows: 1) We propose an efficient network, CSFN, for SISR to focus on channel information and different information fusion, which achieves high performance with low complexity. 2) To further reduce parameters and calculations, we design a new recursive CNN model (CSFN-M) to minimize redundant parameters. 3) Extensive experiments demonstrate that our CSFB can extract information efficiently and treat channels unequally, and after fusing that information, CSFB can provide more efficient feature maps for reconstruction.

2. PROPOSED METHOD

2.1. Network Architecture

As shown in Fig.2, our CSFN is divided into three parts: shallow feature extraction block (FEBlock), residual channel splitting and fusion blocks (CSFBlocks) and upscale block (UPBlock). Denote by x and y the input and output of CSFN. In FEBlock, we use one convolutional layer to extract features from the LR image. F_0 is the shallow features extracted by FEBlock. CSFBlocks consist of multiple CSFBs and a bottom convolution layer, and the output of CSFBlocks can be expressed as

$$F_r = f_t(H_{CSF,n}(\dots(H_{CSF,1}(F_0))\dots)) + F_0 \quad (1)$$

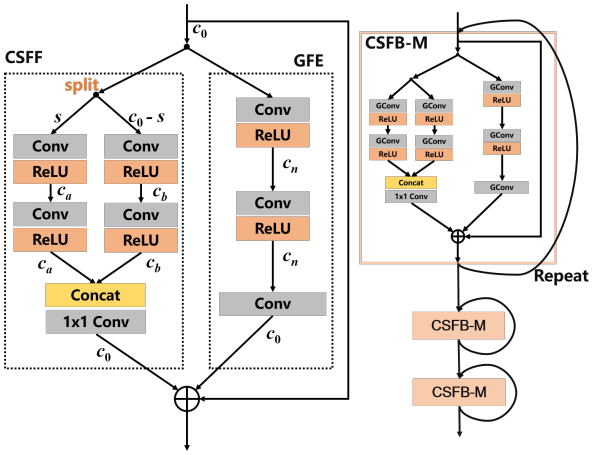

Fig. 3. The architecture of proposed CSFB (left) and CSFB-M (right).

where f_t is the bottom convolution function and $H_{CSF,i}$ represents the operation of i-th CSFB. The detail of $H_{CSF,i}$ can be found in Section 2.2. The final part is UPBlock, which does the LR-HR transforming and reconstruct the HR image. We use ESPCN in our upscale module. For scale 2 and scale 3, we use one Conv-PixelShuffle structure. Two Conv-PixelShuffle modules are used for scale 4.

2.2. Residual Channel Splitting and Fusion Block

Our proposed CSFB (shown in Fig.3) can be roughly divided into two pipelines. The left pipeline is a feature fusion module based on channel split (CSFF) and the right one is a global feature extraction (GFE) module. Let's denote the input feature maps of i-th CSFB as F_{i-1}, and the total number of channels in F_{i-1} is c_0. Then the feature maps F_{i-1} flow into CSFF, GFE and the other part of CSFB for residual learning.

CSFF module mainly focuses on the imbalance of channel information. Since the information extracted from LR images contains various information, the variance of channel features cannot be neglected. To make a distinction between different channel features, we adopt an asymmetric channel split to divide the feature into two parts. As Fig.3 shows, we split F_{i-1} into two parts which contain s and $c_0 - s$ channels respectively, where s is less than $c_0/2$ in our network. We will discuss the purpose of this asymmetric splitting in Section 3.1. We denote the feature maps after splitting operation as $F_{i,s1}$ and $F_{i,s2}$, and the dimensions of left and right branch in CSFF as c_a and c_b, subjected to $c_a + c_b = c_0$. $c_a > s$ is the restriction on c_a to guide the network to extract more information in $F_{i,s1}$, $c_b < c_0 - s$ can be explained as less feature maps in $F_{i,s2}$ would be used in this block. The operations of CSFF can be formulated as:

$$F_{i,s1o} = \tau(f(\tau(f(F_{i,s1}, s, c_a)), c_a, c_a)) \quad (2)$$

$$F_{i,s2o} = \tau(f(\tau(f(F_{i,s2}, c_0 - s, c_b)), c_b, c_b)) \quad (3)$$

where $F_{i,s1o}, F_{i,s2o}$ denote the outputs of two branches, and τ is the ReLU activation function, $f(F, c_a, c_b)$ is the convolutional operation with c_a as input feature dimension and c_b

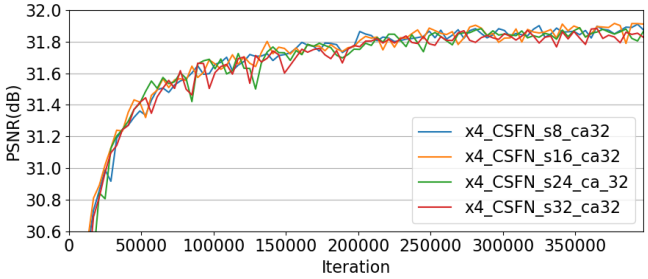

Fig. 4. The performance of different splitting on Set5 with the scaling factor $\times 4$.

as output feature dimension. Then the two feature maps are fused by concatenation and the 1×1 convolution.

The main purpose of GFE module (the right pipeline) is to compensate for CSFF module since CSFF module only uses partial channel information independently which leads to incomplete global information. A channel compression and expansion unit is used to extract features and promote channel information fusion as well as reducing the number of parameters. GFE is composed of 2 Conv-ReLUs and 1 Conv layer. The feature maps dimension of i-th Conv-ReLU output are denoted as c_n, c_n, and c_0 respectively ($c_n < c_0$). This operation can be described as follows:

$$F_{i,L} = f(\tau(f(\tau(f(F_{i-1}, c_0, c_n)), c_n, c_n)), c_n, c_0) \quad (4)$$

The feature maps from CSFF and GFE are then aggregated to get the fusing features and residual learning. Thus, the output of i-th CSFB can be formulated as:

$$F_i = H_{CSFF,i}(F_{i-1}) + H_{GFE,i}(F_{i-1}) + F_{i-1} \quad (5)$$

where $H_{CSFF,i}$, $H_{GFE,i}$ denote the operation collections in i-th CSFF and GFE modules respectively.

To further reduce parameters and enhance feature maps, we propose a more lightweight network(CSFN-M) which replaces CSFB with CSFB-M. As shown in Fig.3 (right), we firstly adopt group convolutional layer in CSFF and GFE, and then the feature maps which flow into CSFB-M would be enhanced for t times, This can be described as follows:

$$F_i = R_i(R_i(\dots R_i(F_{i-1})\dots)) \quad (6)$$

where R_i denotes the operation collections in i-th CSFB-M modules, and there are t R_i in the above formula.

2.3. Implementation Details

In our proposed networks, the kernel sizes in convolutional layers are all set to 3×3 except the specified 1×1 convolutional layers. The number of channels in $F_i (i = 1, 2, ..., n)$ is set to 64. We use 10 CSFBs in our proposed CSFN and 5 for CSFN-M. In each CSFB, c_0, s, c_a, c_b and c_n are set to 64, 16, 32, 32 and 16 respectively. For our proposed CSFB-M, we use the same setting with CSFB but the number of groups is set to 2, t is set to 3 for each CSFB-M.

During the training process, for each batch, 16 LR RGB patches with the size of 48×48 are randomly selected from DIV2K dataset(800 training images). Then data augmentation is applied by flipping horizontally or vertically or rotating 90°. The proposed network is implemented with PyTorch and optimized using Adam with $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning rate is set to 10^{-4} and decreases by a factor of 2 for every 2×10^5 iterations during the total 10^6 iterations. The loss function we used is $L1$ loss. We use Set5, Set14, BSD100, and Urban100 for test and benchmark. The degradation operator we used is bicubic downscaling and the scaling factors are set to $\times2$, $\times3$, and $\times4$. We use two commonly used evaluation metrics in SR task to compare the rebuild quality: peak signal-to-noise ratio (PSNR) and structural similarity (SSIM).

Table 1. Experiments in the ablation study. All models are trained in 1×10^5 iterations with the scaling factor $\times4$.

CSFF	GFE	Params	Set5	Set14	BSD100	Urban100	Average
×	×	484K	31.40	28.11	27.23	25.15	26.43
✓	×	496K	31.44	28.13	27.24	25.18	26.45
×	✓	435K	31.39	28.08	27.21	25.09	26.39
✓	✓	559K	31.58	28.20	27.28	25.26	26.51

3. EXPERIMENTS

3.1. Network Analysis

The effect of CSFB. To investigate the effect of asymmetric splitting, we train our model roughly with different s in CSFF for total $4\times$ 10^5 iterations. Specifically, we set the values of s to 8, 16, 24, and 32 respectively, and 5 CSFBs are used in this experiments, thus the four models have the same number of parameters, other settings are the same with Section 2.3. Fig.4 shows that appropriate asymmetric splitting can generate 0.07dB improvements compare with the symmetric splitting. We then adopt the ablation study to investigate the effects of CSFF and GFE in our CSFN. For better comparison and minimize the performance impact of rising parameter quantities, we only use 3 CSFBs to build a very shallow network. A ReLU-Conv module is used to replace the block in CSFB when there are no CSFF and GFE blocks in the network. Table 1 presents CSFF has slight improvement since it can not get global information from channels, GFE has worse performance with fewer parameters. After combining the CSFF and GFE, the performance would increase a lot compared with only using CSFF or GFE.

Since CSFF in our proposed CSFB plays a key role in the whole model, we then inspect the weights of the last 1x1 Conv layer in CSFF to explore the channel fusion of the two branches. The model we used is a trained $\times2$ model. Note that $c_0 = 64$ and $c_a = c_b = 32$, that is one output channel in CSFF is generated by left part of CSFF(CSFF-L, c_a) and right part of CSFF(CSFF-R, c_b), we than calculate the average weight(absolute value) of CSFF-L and CSFF-R and compute the weight ratio of CSFF-L to all channels. Fig. 5 shows that the first 16 channels use CSFF-L features more in all blocks because those feature maps would be used in the next CSFF-L, thus some feature map channels are constantly being strengthened during this process by using those features to build more features. Fig. 6 is the average(absolute value)

Table 2. Average PSNR/SSIMs for scale $\times 2$, $\times 3$ and $\times 4$. Red/blue text: best/second-best, underline text: best result below 500K parameters.

Scale	Dataset/Params	DRRN[20]	CARN-M[21]	SRFBN-S[24]	**CSFN-M**	VDSR[10]	Memnet[25]	IDN[26]	CARN[21]	**CSFN**
x2	Params/FLOPs	297K/6802.9G	412K/91.2G	282K/679.7G	**298K/147.1G**	665K/613.7G	677K/623.9G	551K/123.5G	1592K/222.8G	**845K/195.9G**
	Set5	37.74/0.9591	37.53/0.9583	37.78/0.9597	**37.72/0.9596**	37.53/0.9587	37.78/0.9597	37.83/0.9600	37.76/0.9590	**37.85/0.9600**
	Set14	33.23/0.9136	33.26/0.9141	33.35/0.9156	**33.32/0.9155**	33.03/0.9124	33.28/0.9143	33.30/0.9148	33.52/0.9166	**33.54/0.9187**
	BSD100	32.05/0.8973	31.92/0.8960	32.00/0.8970	**32.01/0.8980**	31.90/0.8960	32.08/0.8978	32.08/0.8985	32.09/0.8978	**32.10/0.8990**
	Urban100	31.23/0.9188	31.23/0.9193	31.41/0.9207	**31.47/0.9221**	30.76/0.9140	31.31/0.9195	31.27/0.9196	31.92/0.9256	**31.88/0.9257**
x3	Params/FLOPs	297K/6802.9G	412K/46.6G	376K/828.1G	**391K/75.6G**	665K/613.7G	677K/623.9G	553K/54.8G	1592K/119.1G	**1030K/106.7G**
	Set5	34.03/0.9244	33.99/0.9236	34.20/0.9255	**34.11/0.9250**	33.66/0.9213	34.09/0.9248	34.11/0.9253	34.29/0.9255	**34.31/0.9267**
	Set14	29.96/0.8349	30.08/0.8367	30.10/0.8372	**30.13/0.8383**	29.77/0.8314	30.00/0.8350	29.99/0.8354	30.29/0.8407	**30.28/0.8406**
	BSD100	28.95/0.8004	28.91/0.8000	28.96/0.8010	**28.96/0.8018**	28.82/0.7976	28.96/0.8001	28.95/0.8013	29.06/0.8034	**29.08/0.8043**
	Urban100	27.53/0.8378	27.55/0.8385	27.66/0.8415	**27.69/0.8420**	27.14/0.8279	27.56/0.8376	27.42/0.8359	28.06/0.8493	**28.06/0.8497**
x4	Params/FLOPs	297K/6802.9G	412K/32.9G	483K/1037.3G	**372K/55.0G**	665K/613.7G	677K/623.9G	555K/30.9G	1592K/91.2G	**993K/84.2G**
	Set5	31.68/0.8888	31.92/0.8903	31.98/0.8923	**31.90/0.8912**	31.35/0.8838	31.74/0.8893	31.82/0.8903	32.13/0.8937	**32.10/0.8935**
	Set14	28.21/0.7720	28.42/0.7762	28.45/0.7779	**28.42/0.7773**	28.01/0.7674	28.26/0.7723	28.25/0.7730	28.60/0.7806	**28.61/0.7806**
	BSD100	27.38/0.7284	27.44/0.7304	27.44/0.7313	**27.45/0.7316**	27.29/0.7251	27.40/0.7281	27.41/0.7297	27.58/0.7349	**27.60/0.7367**
	Urban100	25.44/0.7638	25.62/0.7694	25.71/0.7719	**25.68/0.7718**	25.18/0.7524	25.50/0.7630	25.41/0.7632	26.07/0.7837	**26.10/0.7866**

Table 3. Performance of $\times 4$ SR by CSFB and RCAB in terms of PSNR(dB). All models are trained in 4×10^5 iterations.

Model	Params	Set5	Set14	BSD100	Urban100	Average
Baseline	742K	31.81	28.42	27.43	25.65	26.78
RCAB	745K	31.91	28.40	27.42	25.69	26.80
CSFB	743K	31.95	28.46	27.46	25.73	26.84

output feature map, the features in CSFF-L have much clear outline than CSFF-R whether in the shallow or deep layer, which proves the CSFF module takes full use of the input channel by splitting channels and enhances the area which is difficult to rebuild. Fig. 6 also shows that feature maps in GFE are smoother than CSFF despite the network depth, and the element-wise "Add" results (last col) have more clear contour profile than CSFF and GFE, which indicates that the GFE module can enhance the final results. So those can verify the capability of CSFB for extracting useful features to reconstruct images.

Comparison with channel attention. To validate the effectiveness of our CSFB, we compare it with RCAN's RCAB[16], which is the typical channel attention (CA) mechanism used in SR task. The baseline model we used is EDSR's residual block with 64 channels. For a fair comparison, we set $c_n = 25$ to ensure that CSFB and residual block have similar parameter quantities. 5 CSFBs are used in this experiment. Table 3 shows that our proposed CSFB could obtain higher performance with fewer parameters than RCAB for SR tasks.

3.2. Comparison with State-of-the-Art Models

We compare our method with some state-of-the-art methods: VDSR [10], DRRN [20], MemNet [25], IDN [26], CARN [21] and SRFBN [24]. For a fair comparison, the heavy network such as EDSR, DBPN, RDN, RCAN is excluded. To better reveal the computational complexity of each model, we calculate the floating-point operations (FLOPs) under the assumption that the size of the output image is 1280×720. Table 2 shows our CSFN has better performance with fewer parameters and FLOPs than CARN, note that CARN uses larger patch sizes in training time and multi-scale training strategy to improve the final result which would significantly increase

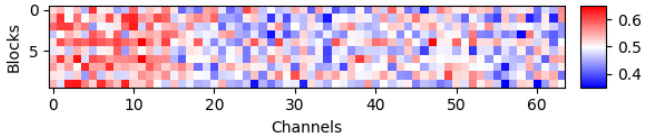

Fig. 5. The weight ratio of CSFF-L in output channels.

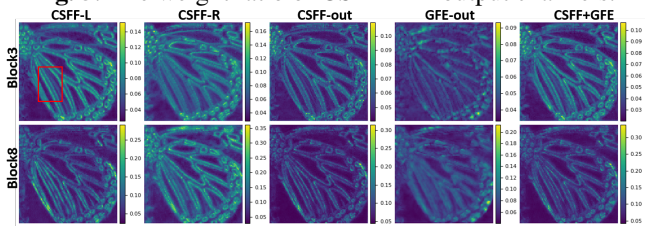

Fig. 6. The average feature map in different layer.

the training time. For a more lightweight model, our CSFN-M can achieve higher performance compared with DRRN, CARN-M, and similar or better performance than SRFBN-S, but SRFBN-S has larger FLOPs. Fig. 1 shows the visual comparisons. The proposed CSFN and CSFN-M rebuild the grass edge more accurately while other models only smoothes the area in "img076" from Urban100. The results on image "barbara" show that other models generate the wrong texture when our CSFN can predict the texture of this spotted cloth correctly, and CSFN-M can correctly predict partial structures with fewer parameters. In "img092" from Urban100, all other methods infer the wrong black line, but our CSFN can make full use of the information in low-resolution images to accurately estimate the direction of the line.

4. CONCLUSION

In this paper, we propose a novel low complexity residual channel splitting and fusing network (CSFN) architecture which takes into account the unevenness in feature map channels and can extract features more efficiently for SISR task. Our proposed method can reconstruct complex objects accurately and show superior results considering the PSNR and the SSIM. This network will be meaningful for practical application. In the future, this work will be used to image restoration and video super-resolution tasks.

5. REFERENCES

[1] S. Dai, M. Han, W. Xu, Y. Wu, and Y. Gong, "Soft edge smoothness prior for alpha channel super resolution," in *CVPR*. IEEE, 2007.

[2] H. A. Aly and E. Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1647–1659, 2005.

[3] Q. Shan, Z. Li, J. Jia, and C.-K. Tang, "Fast image/video upsampling," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5, pp. 153, 2008.

[4] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Computer graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[5] T.-M. Chan, J. Zhang, J. Pu, and H. Huang, "Neighbor embedding based super-resolution algorithm through edge detection and feature selection," *Pattern Recognition Letters*, vol. 30, no. 5, pp. 494–502, 2009.

[6] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[7] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE TPAMI*, vol. 38, no. 2, pp. 295–307, 2016.

[8] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network.," in *CVPR*, 2017.

[9] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *The European Conference on Computer Vision Workshops (ECCVW)*, September 2018.

[10] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *CVPR*, 2016, pp. 1646–1654.

[11] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *ECCV*. Springer, 2016, pp. 391–407.

[12] 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 subpixel convolutional neural network," in *CVPR*, 2016, pp. 1874–1883.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[14] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks.," in *CVPR*, 2017.

[15] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *CVPRW*, 2017.

[16] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhang, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," *ECCV*, 2018.

[17] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *ICCV*. IEEE, 2017, pp. 4809–4817.

[18] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu, "Residual dense network for image super-resolution," in *CVPR*, 2018.

[19] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *CVPR*, 2016, pp. 1637–1645.

[20] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *CVPR*, 2017.

[21] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," *arXiv preprint arXiv:1803.08664*, 2018.

[22] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, Jixiang Li, and Qingyuan Li, "Fast, accurate and lightweight super-resolution with neural architecture search," *CoRR*, vol. abs/1901.07261, 2019.

[23] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," 2018.

[24] Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu, "Feedback network for image super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *CVPR*, 2017, pp. 4539–4547.

[26] Z. Hui, X. Wang, and X. Gao, "Fast and accurate single image super-resolution via information distillation network," in *CVPR*, 2018, pp. 723–731.