On the Sparsity of Image Super-resolution Network

Chenyu Dong[1,2], Hailong Ma[1,2], Jinjin Gu[3], Ruofan Zhang[1,2], Jieming Li[2] and Chun Yuan[1,4,*]

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University
[2]Huawei Technologies
[3]The University of Sydney
[4]Peng Cheng National Laboratory

Abstract

The over-parameterization of neural networks has been widely considered for a long time. This allows us to find sub-networks that can improve the parameter efficiency of neural networks from over-parameterized networks. In our study, we used EDSR (25) as the backbone network to explore the parameter efficiency in super-resolution(SR) networks in the form of sparsity. Specifically, we search for sparse sub-networks at the two granularity of weight and kernel through various methods and analyze the relationship between the structure and performance of the sub-networks. In summary, our work: (1) On weight granularity, we observe the "Lottery Ticket Hypothesis" (10) from a new perspective in the regression task of SR; (2) On convolution kernel granularity, we apply several methods to explore the influence of different sparse sub-networks on network performance and found that based on certain rules, the performance of different sub-networks rarely depends on their structures; (3) We propose a very convenient width-sparsity method on convolution kernel granularity, which can improve the parameter utilization efficiency of most SR networks.

1 Introduction

Super-resolution (SR) networks aim at predicting high-resolution (HR) images from low-resolution (LR) observations. SR networks often contain a large number of parameters. Obtaining better performance with fewer parameters is an important research topic. Network sparsification and network pruning are considered to be promising methods. The success of the "Lottery ticket hypothesis" (LTH) (10) seems to point in a surprising direction. LTH uses a simple iterative pruning method to find a sparse sub-network with better performance than the densely connected network. In this work, we try to obtain a more efficient SR network substructure by building a sparse SR network. However, finding "lottery tickets" for SR networks is difficult. In other words, there is a positive correlation between the parameters of the SR networks and the performance.

We first conduct our research at the weight level sparsity – we try to find a sparse subnetwork that is comparable to the original network when trained in isolation. This is also called the "winning lottery ticket" (10) for SR networks. Our experimental results show that a sparse connection will lead to performance degradation when an SR network is well-trained. While the "winning ticket phenomenon" only exists in the networks that are not well-trained, and can not generate comparable performance to the original network. In other words, we can not find a sparse sub-network from a dense-large SR network without experiencing a performance drop as Lottery Ticket Hypothesis described.

*Corresponding author.

I Can't Believe It's Not Better Workshop at NeurIPS 2022.

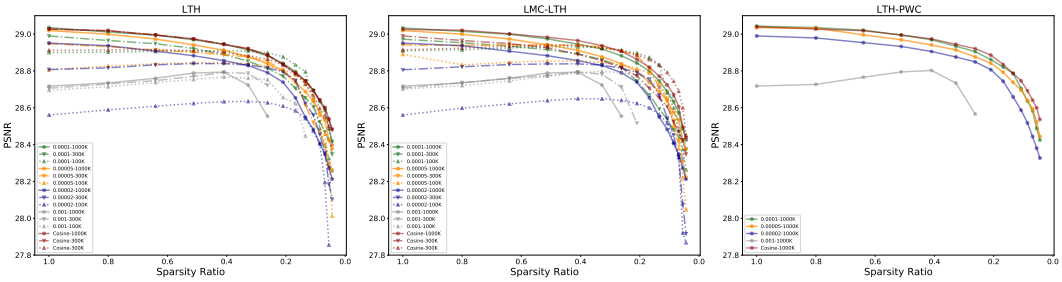
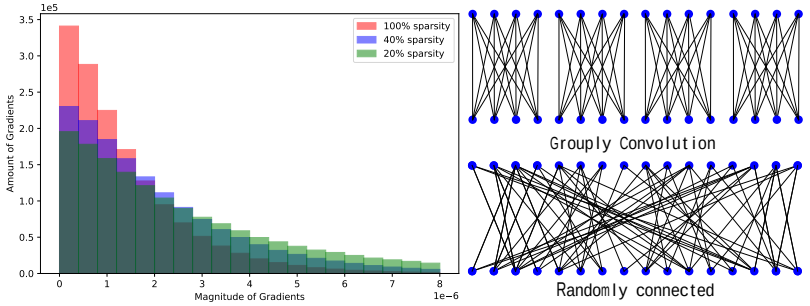


Figure 1: Experimental results of three weight-level methods on DIV2k_val dataset. The lines of different colors represent the results of different learning rates(0.001, 0.0001, 0.00005, 0.00002, CosineLR). In figure (a) and (b), the "solid lines", "dash-dotted lines", and "dashed lines" denotes the experiment with the iteration number of 1,000K, 300K, and 100K, respectively.



(a) (b)

Figure 2: (a) Histogram of gradient distribution under different sparsity at 1,000K iteration, all layers in EDSR are collected. (b) Example of Group convolution and Randomly-Connected neural network layers. The blue dots can be considered as channels in the convolution layer, Black lines can be considered as convolution kernels in convolution layers.

Secondly, we use four methods to sparse the network at the convolution kernel granularity, which can be abstracted as pruning off a part of the connections between channels. Our experiments show that all these methods of obtaining sub-networks, whether they are reasonable or not, have obtained similar performance under the same width and sparsity (the difference in PSNR does not exceed 0.03dB) unless transmission of information is blocked in some channels. Though we can't find a sparse sub-network comparable to the fully-connected network, we implement a very convenient width-sparsity trade-off method that can optimize the parameter efficiency of SR networks. According to our experimental results at the kernel level, we can conveniently define a large-sparse network structure in the initialization phase. In this way, we can improve the network's performance (PSNR improved 0.04dB-0.08dB on Div2k_val) without changing the FLOPS or reducing parameters without changing the network's performance. Such a method only needs to decorate the convolution layer without changing the network infrastructure or training strategy. Moreover, the network structure can be defined at the initialisation stage, saving computational expenses in the pruning process.

In summary, our work: (1) Apply several methods to the EDSR network, and we could not find a sparse sub-network that can perform comparably to the original network like the lottery ticket hypothesis; (2) Draws an important observation that the performance of sparse SR networks with the same width and sparsity are surprisingly similar at the kernel level unless the transmission of information between channels is blocked; (3) We propose a very convenient method that can be deployed on most existing super-resolution networks to improve parameter efficiency.

2 Weight Level Sparsity

In this part of the experiment, we use three methods based on the lottery hypothesis to obtain sub-networks, and have a new understanding of the lottery ticket hypothesis from the perspective of the regression task like super-resolution.

2.1 Obtain sub-network

Weight is the smallest unit in a neural network; therefore, sparsity at the weight level can minimize the performance drop of networks. The LTH (10) is one of the most influential methods in recent years. The LTH proposes that in classification tasks, we are able to find sparse sub-networks from a randomly-initialized dense network that this sub-network can be trained from scratch in isolation and reach test accuracy comparable to the original network. Specifically, in LTH, we first initialize a classification network randomly with parameters θ_0. And then, we train the network for T iterations and arrive at θ_T. The third, we use a mask $m \in \{0,1\}^{|\theta|}$ to prune $p\%$ of the existing parameters in θ_T. At last, we reset the remaining parameters to their initialized value θ_0, apply m to θ_0 and convert parameters to $m \odot \theta_0$. Then, the algorithm will repeat the above steps in the way of iterative pruning for several rounds. Through this pruning method, a sparse sub-network comparable to the fully-connected network can be found in the "winning ticket" classification task. Besides, LMC-LTH (11) proposes to rewind parameters to a certain time after a small amount of training θ_t, instead of the initialization stage θ_0. Liu et al. (27) found that replacing then parameters rewind operation in LTH with a fine-tune process for another T-epoch can obtain a better sub-network compared with LTH. We chose these three representative methods to obtain sub-networks and specific experimental settings will be described in detail in Appendix.

2.2 Results

Our experimental results are shown in Figure 1. Figures (a), (b), and (c) show the results of LTH, LMC-LTH, and LTH-PWC respectively, where the solid lines represent the experimental results under different learning rates with 1000K iteration. Although these results differ slightly in numerical value, these curves have similar trends. We can see that in the three groups of experiments with 1,000K iterations, except for the experiment where the learning rate is 1×10^{-3}, the network performance decreases with the decrease of sparsity. The first experimental phenomenon we draw is that, different from the classification task, even if we set the learning rate (2×10^{-5}) to a value much smaller than the normal case (2×10^{-4}), when the network is sufficiently trained, we can not find a "winning ticket" in SR network ("winning ticket" usually appears in the classification task when the learning rate is relatively small). On the contrary, when the learning rate is set too large (1×10^{-3}), a sub-network with better performance than the original network appears. But we also noticed that there was a significant performance gap between the network at 1×10^{-3} learning rate and the appropriate learning rate when fully connected.

In figures (a) and (b), dash-dotted lines and dotted lines show the experimental results with the iteration of 300K and 100k, respectively. The results of these two methods still have a similar phenomenon. It can be noticed that when the number of iterations is 300K, the "winning ticket" appears in the network with a learning rate of 2×10^{-5}. And when the number of iterations is further reduced and set to 100k, there are "winning tickets" under all learning rates. We note that though the reasons for winning tickets' appearance are different from those before (1×10^{-3} learning rate), the same phenomenon is that the network has not reached its own capacity of expressiveness. Specifically, there will be a gap of more than 0.2dB in PSNR.

We want to emphasize that super-resolution is essentially a complex regression task, therefore, there will be no obvious overfitting phenomenon like the classification task in the training process. This makes the model always in the convergence stage during the SR network training process. In other words, the SR network will continue to improve the performance in the later stage of training even with a small learning rate. While in the classification task with a low learning rate, the network may converge to the local optimal solution. To verify our idea, we compare the gradient distribution of the network with a small learning rate under different sparsity. It can be seen from the Figure 2a that the sparser the network, the smaller the gradient.

In conclusion, we tend to think that the sparse network changes the solution space of the network by constraining the weights and gradients, the change of gradient makes the network jump out of the local optimal (usually appears under the small learning rate).

According to the experimental results above, we can draw two novel conclusions about Lottery Ticket Hypothesis in the super-resolution task: (1) Different from classification tasks, in a sufficiently trained SR network, even if the learning rate is set much smaller than the normal case, the sparsity of the network will lead to performance degradation, and the higher the sparsity degree, the greater the

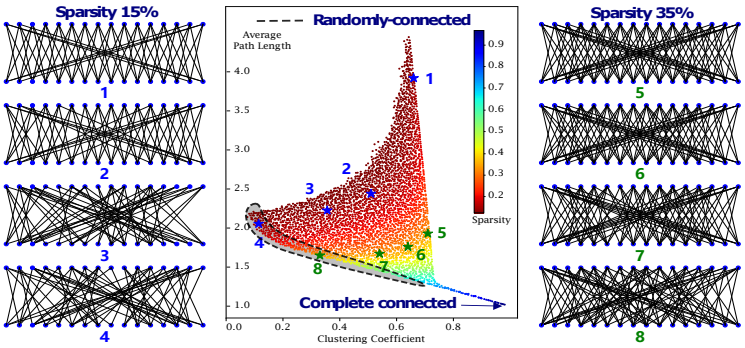
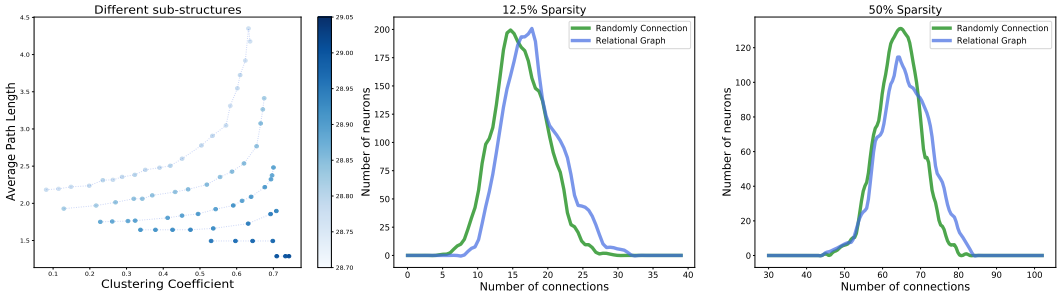


Figure 3: 64-channel convolutional layers' substructures generated by the WS-flex graph generator. We show the sparsity of different substructures in the form of a heat map and select eight substructures for visual display and mark their positions in the heat map.



(a) (b)

Figure 4: (a) Experimental results of 67 different substructures under six sparsity degrees (12.5%, 18%, 25%, 35%, 50%, 70%) on DIV2k_val dataset. We show the PSNR of each substructure after training in the form of a heat map. The points on each dotted line have the same sparsity. (b) Distribution of the number of neuronal connections in random connections and relational graph connections. The x-axis represents the total number of connections of a node, The y-axis represents the number of nodes with x connections.

performance degradation. (2) When the SR network does not reach its expression ability through training, a fake "winning lottery" will appear in the process of network pruning.

3 Kernel level sparsity

In this section, we conducted experiments at the kernel level sparsity. Although we have obtained similar experimental results at weight level (sparsity will lead to the decline of network performance), we are also surprised to find that in an image super-resolution network, the performance of the network almost only depends on the width and sparsity of the network rather than the structures.

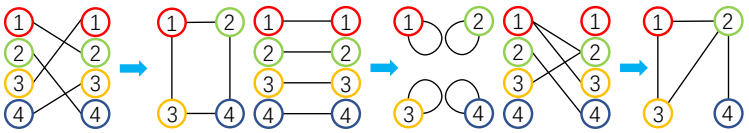
3.1 Obtain sub-network

We describe the obtaining of sparse sub-networks under the following unified framework. Consider an SR network $f(I^{LR}; \Omega, \Theta)$ parameterized by convolution weights for L layers $\Omega = \{\omega_1, \ldots, \omega_L\}$ and all the other parameters Θ with an input LR image I^{LR}.

Then, a sub-network is represented as

$$f(I^{LR}; \{m_1 \odot \omega_1 \ldots, m_L \odot \omega_L\}, \Theta), \tag{1}$$

where $m_1, \ldots, m_L$ represent L masks for every layers, $\odot$ indicates the masking operation with broadcasting. Suppose a weight ω_l is a tensor of size $C_{\text{in}} \times C_{\text{out}} \times K \times K$, where C_{in} and C_{out} denote the number of input and output channels, and K is the kernel size. Then m_l is a binary mask of size $C_{\text{in}} \times C_{\text{out}}$, where each element indicates whether the corresponding kernel is masked out. In this way, the masks obtained through different methods indicate different sparse sub-networks. The sparsity of the obtained sub-network can be approximated by $\frac{\sum_i^{C_{\text{in}}} \sum_j^{C_{\text{out}}} m_{i,j}}{C_{\text{in}} \times C_{\text{out}}}$. For the global



(a) (b) (c)

Figure 5: Mapping of four channels' convolution layer connection structure in the relational graph. (a) can represent the relational graph, (b) can represent the group convolution, and (c) can represent the random connection.

Sparsity	Random		Group		ERSS-G		ERSS-L		Graph-1		Graph-2	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
12.5%	28.64	0.8102	28.70	0.8115	28.77	0.8140	28.75	0.8133	28.79	0.8144	28.78	0.8143
15%	28.71	0.8121	-	-	28.80	0.8144	28.79	0.8142	28.81	0.8147	28.81	0.8147
25%	28.78	0.8141	28.82	0.8151	28.88	0.8164	28.86	0.8162	28.88	0.8166	28.89	0.8169
35%	28.86	0.8161	-	-	28.94	0.8181	28.93	0.8180	28.91	0.8175	28.93	0.8180
50%	28.92	0.8178	28.93	0.8180	28.99	0.8196	28.97	0.8189	28.96	0.8188	28.97	0.8190
70%	28.96	0.8186	-	-	29.01	0.8201	29.00	0.8196	29.00	0.8197	29.01	0.8200

Table 1: Quantitative results of different sparse sub-network obtained by different methods on the DIV2k_val dataset. "Graph-1" denotes relational graphs with low Average Path Length and "Graph-2" denotes relational graphs with high Average Path Length. "ERSS-L" denotes layer-wise pruning, and ERSS-G denotes global pruning.

unified operations, we omit the subscript l without loss of generality. Next, we will introduce the mask acquisition methods involved in this part.

Grouped and randomly-connected convolutions. These two are the most straightforward and simple methods. We show examples for these two kinds of sub-networks in Figure 2b. For a grouped convolution layer with G groups, we assume that the input has N channels. Then this operation is to first divide N channels into G parts. Each group corresponds to $\frac{N}{G}$ channels, which are independently connected within each group. After each group convolution is calculated, the output is concatenated as the output of this layer. For a grouped convolution layer with G groups, only $\frac{1}{G}$ of all elements in the mask are set to 1, thus the sparsity is equal to $\frac{1}{G}$.

If grouped convolution is the most regular way to obtain the mask, then randomly-connected convolution is the other extreme. Given a pre-defined sparsity q, then each element in the mask $m_{i,j}$ follows a Bernoulli distribution $P(m_{i,j} = 1) = q$. This is the simplest way to achieve a certain sparsity in a convolution layer.

Network pruning. Generally, this kind of method starts with training a large, over-parameterized network, and then proposes to remove redundant parameters without significantly compromising accuracy. Mao et al. (30) have taken the sum of the absolute values of all weights in a 2-D convolution kernel as a unit, and then prune part of kernels with the lowest-magnitude. In our study, we will use this simplest way to do iterative pruning for 6 rounds (corresponding to six sparsity degrees). For this method, we will apply global pruning (pruning the the lowest-magnitude kernels collectively across all convolutional layers) and layer-wise pruning (pruning each layer separately at the same rate).

Networks guided by relational graphs. You et al. (41) have described a novel approach to represent the complex and diverse possible sparse connections between layers in a neural network. The relational graph depicts the information interaction between the input channels and the output channels, and a graph $\mathcal{G} = (\mathcal{V}, \mathcal{C})$ is used to represent this interaction, where $\mathcal{V}$ is the channel set and $\mathcal{C}$ contains connection between channels. The graph measures of $\mathcal{G}$ can characterize important properties for the corresponding architecture. You et al. (41) have presented that average path length and clustering coefficient are two important measures. Through the WS-flex graph generator (39), we can sample graphs with different average path lengths and clustering coefficients as much as possible. Figure 3 shows the distribution of generated graphs and some network connection examples. It is worth noting that the architectures obtained through random connection only account for a small part of all possible architectures, and we mark them by the gray area in Figure 3. With the above correspondence between the relational graphs and the network architectures, we can discover various other possible architectures that cannot be simply obtained by the other methods.

3.2 Results

Table 1 show the experimental results of different methods. To our surprise, the substructure generated by the relational graph can achieve a result comparable to the pruning method. And these two methods perform better than group convolution and random connection, and the more sparse the network, the greater the impact of its connection structure on the performance of the network.

In order to further study the relationship between the SR network's structures and performance, we selected six sparsity degrees (12.5%, 18%, 25%, 35%, 50%, 70%), and uniformly sampled from Figure 3 according to these six sparsity degrees. Figure 4a shows our experimental results. In this figure, the PSNR is indicated by the color depth. It can be found that even if our sampling under the same sparsity covers a large range, the performance between the substructures generated by the relational graph is almost equivalent. This shows to a certain extent that for an SR network, the connection structure of the network at the kernel level has little impact on the model performance.

The difference between the performance of randomly (group) connection and relational graph connection is that the substructure sampled from the graph is a "connected structure". Figure 5 shows three kinds of graph structure mapping. It can be seen that although the three structures have the same number of connections between the four channels, only the mapping of structure (a) is a connected graph. Such a structure ensures that each node in the network can directly or indirectly complete information transmission (Receive or send) with any other node in the network. In other words, the connected graph not only ensures smooth information transmission but also ensures global information transmission. Figure 5 shows the node connection distribution of random connection and relational graph connection under two sparsity degrees. It can be seen that the relationship graph suppresses the number of nodes with fewer connections at low sparsity through the principle of "connected graph". When the sparsity increases, this repression is weakened, and the performance gap between the two connection methods is also reduced.

In summary, in this part of the study, we make a very interesting discovery: When the relational graph structure of an SR network can be mapped to a "connected graph", its performance depends almost entirely on its channel number and sparsity, and rarely on its structure. At the same time, such a substructure has an expression capacity comparable to that of the pruning method.

4 Width-sparsity trade off

Zhu et al. (49) have proposed that on weight level, pruned models (large-sparse) can outperform their smaller, but dense (small-dense) counterparts with identical FLOPS. This is a simple method to improve the performance of most existing models in the classification task, but a pruning process is still needed to obtain a superior sub-network. Moreover, although we all know that sparsity on the weight level can better maintain network performance than on the kernel level, our experimental results show that fine-grained sparsity on large SR networks will lead the model difficult to converge. At the same time, the model is prone to collapse on networks with an attention mechanism (more details in appendix). Based on these phenomena above and our experimental results on the kernel level, we hope to obtain a large sparse network more conveniently on the kernel level.

According to our conclusion obtained on kernel-level: when the relational graph of the network structure can be mapped to a connected graph, we can define an approximate optimal sub-network in the initialization phase, which means we can save the computational cost of the pruning process. Meanwhile, sparsity on the kernel level means that compared with the weight level, we can use a coarser granularity to sparse the network, which is helpful for the convergence of the network

Combined with the experimental results on two levels, we propose the following methods to improve the parameter utilization efficiency of the network: (1) Expand the channels of the existing SR network (1.5-2.5 times the original); (2) Sparse the network according to the principle of "a connected graph"; (3) Keep the training settings of the original model unchanged and conduct training.

We apply our method to EDSR (25), PAN (47), IMDN (16) and Lattice-net (29), Experimental results are shown in Appendix. We are pleasantly surprised to find that our method can work on a small network with only 270k parameters such as PAN. We divide the sparse network into performance and efficient modes. The efficient mode can reduce flops by 20% - 40% while maintaining the original performance of the network, performance mode can improve the PSNR on the Div2k_val set by 0.04dB-0.08dB while maintaining the FLOPS of the original network.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017. 9

[2] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 252–268, 2018. 9

[3] Yue Bai, Huan Wang, ZHIQIANG TAO, Kunpeng Li, and Yun Fu. Dual lottery ticket hypothesis. In *International Conference on Learning Representations*, 2021. 9

[4] Haoyu Chen, Jinjin Gu, and Zhi Zhang. Attention in attention network for image super-resolution. *arXiv preprint arXiv:2104.09497*, 2021. 9

[5] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021. 9

[6] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11065–11074, 2019. 9

[7] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015. 9

[8] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pages 391–407. Springer, 2016. 9

[9] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019. 9

[10] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. 1, 3, 9, 11

[11] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR, 13–18 Jul 2020. 3, 9

[12] Jinjin Gu, Hannan Lu, Wangmeng Zuo, and Chao Dong. Blind super-resolution with iterative kernel correction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1604–1613, 2019. 9

[13] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3012–3021, 2020. 9

[14] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1387–1395, 2016. 9

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 9

[16] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2024–2032, 2019. 6

[17] Xinrui Jiang, Nannan Wang, Jingwei Xin, Xiaobo Xia, Xi Yang, and Xinbo Gao. Learning lightweight super-resolution networks with weight pruning. *Neural Networks*, 144:21–32, 2021. 9

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016. 9

[19] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645, 2016. 9

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 9

[21] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 9

[22] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Exploiting kernel sparsity and entropy for interpretable cnn compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2800–2809, 2019. 9

[23] Zheyuan Li, Yingqi Liu, Xiangyu Chen, Haoming Cai, Jinjin Gu, Yu Qiao, and Chao Dong. Blueprint separable residual network for efficient image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 833–843, 2022. 9

[24] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021. 9

[25] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 1, 6, 9, 10, 11

[26] Chen Lin, Zhao Zhong, Wu Wei, and Junjie Yan. Synaptic strength for convolutional neural network. *Advances in Neural Information Processing Systems*, 31, 2018. 9

[27] Ning Liu, Geng Yuan, Zhengping Che, Xuan Shen, Xiaolong Ma, Qing Jin, Jian Ren, Jian Tang, Sijia Liu, and Yanzhi Wang. Lottery ticket preserves weight correlation: Is it desirable or not? In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7011–7020. PMLR, 18–24 Jul 2021. 3, 9

[28] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 9

[29] Xiaotong Luo, Yuan Xie, Yulun Zhang, Yanyun Qu, Cuihua Li, and Yun Fu. Latticenet: Towards lightweight image super-resolution with lattice block. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 272–289. Springer, 2020. 6

[30] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J Dally. Exploring the granularity of sparsity in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 13–20, 2017. 5

[31] Yiqun Mei, Yuchen Fan, Yuqian Zhou, Lichao Huang, Thomas S Huang, and Honghui Shi. Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5690–5699, 2020. 9

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 9

[33] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 9

[34] Dehua Song, Chang Xu, Xu Jia, Yiyi Chen, Chunjing Xu, and Yunhe Wang. Efficient residual dense block search for image super-resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12007–12014, 2020. 9

[35] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3147–3155, 2017. 9

[36] Longguang Wang, Xiaoyu Dong, Yingqian Wang, Xinyi Ying, Zaiping Lin, Wei An, and Yulan Guo. Exploring sparsity in image super-resolution for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4917–4926, 2021. 9

[37] Xintao Wang, Ke Yu, Kelvin C.K. Chan, Chao Dong, and Chen Change Loy. BasicSR: Open source image and video restoration toolbox. https://github.com/xinntao/BasicSR, 2020. 9

[38] 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 *European Conference on Computer Vision*, pages 63–79. Springer, 2018. 9

[39] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998. 5

[40] Deyun Wei and Zhaowu Wang. Multi-scale channel network based on filer pruning for image super-resolution. *Optik*, 236:166641, 2021. 9

[41] Jiaxuan You, Jure Leskovec, Kaiming He, and Saining Xie. Graph structure of neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10881–10891. PMLR, 13–18 Jul 2020. 5, 9

[42] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018. 9

[43] Yulun Zhang, Kunpeng Li, Kai Li, Bineng Zhong, and Yun Fu. Residual non-local attention networks for image restoration. *arXiv preprint arXiv:1903.10082*, 2019. 9

[44] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018. 9

[45] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Aligned structured sparsity learning for efficient image super-resolution. *Advances in Neural Information Processing Systems*, 34:2695–2706, 2021. 9

[46] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Learning efficient image super-resolution networks via structure-regularized pruning. In *International Conference on Learning Representations*, 2021. 9

[47] Hengyuan Zhao, Xiangtao Kong, Jingwen He, Yu Qiao, and Chao Dong. Efficient image super-resolution using pixel attention. In *European Conference on Computer Vision Workshop (ECCVW)*. Springer, 2020. 6, 9, 10

[48] Shaochen Zhong, Guanqun Zhang, Ningjia Huang, and Shuai Xu. Revisit kernel pruning with lottery regulated grouped convolutions. In *International Conference on Learning Representations*, 2022. 9

[49] Michael H Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. 2018. 6

A Appendix

A.1 Related Work

A.1.1 Image Super-Resolution

Since Dong (7) introduce the first SR network, plenty of deep learning based methods have been proposed for SR, deeper (8; 18) and wider networks (25), different network strategies (19; 35; 12; 13), residual or dense connections (21; 44; 38), attention mechanism (42; 6; 47; 4), non-local operations (43; 31), transformers (24; 5). In order to make the SR network more efficient and meet the needs of algorithm miniaturization, designing a lightweight SR model is also the focus of both the research community and industry. Early works directly achieved the effect of adjusting the computational complexity by adjusting the scale (depth and width) of the network (8). Other works include introducing less complex operations to improve the efficiency of the network (33; 2; 47), adding complex operations selectively at important locations (4), employing network architecture search (9) technology to find efficient architectures (34). Zhang et al. (45; 46) lighter the specific network by pruning on filter-level. More related to this work, CARN, (2) BSRN (23) introduces group convolution into SR networks to reduce the computational cost. Wang (36) adopts dynamic unstructured pruning to find efficient sparse sub-networks of specific networks. To the best of our knowledge, no other work analyses the utilization efficiency of parameters from the perspective of sparse structure and found a promotion method that can be generalized on most networks.

A.1.2 Sparse Sub-network

Obtaining sparse sub-networks for efficient inference follows a long line of works. Network compression/sparseness can be generally categorized into unstructured and structured methods. Unstructured methods (15; 14) directly pop unimportant weights, but the obtained sparse network can only show computational efficiency when using specialized libraries or hardware. On the contrary, the structured methods trims the entire channel or filter of little importance, so the actual acceleration can be easily achieved without the need for a special accelerator. Kernel-level pruning is a pruning method between structured pruning and unstructured pruning. This kind of pruning method prunes the convolution kernel of 2D dimension as a unit and has attracted more attention in recent years (26; 22; 48).

For unstructured weight pruning, since the lottery ticket hypothesis (10) was proposed, a lot of follow-up work (11; 41; 3) for classification networks has been devoted to finding a sub-network that performs as well as the fully-connected network. We naturally raise a question: can SR networks play lottery tickets? In other words, can we find a sparse sub-network from a dense large SR network without experiencing any performance drop? In addition, in order to obtain more sparse sub-network on different levels, we also conducted sparse experiments on the kernel-level. We note that there are works that try to apply network pruning or network compression technologies to SR networks. However, these methods usually require modification of the network (40; 17) and lose generality.

A.2 Implementations of the main body

A.2.1 Weight Level

We use EDSR(25) as the backbone large network. EDSR only contains residual blocks, no special modules or layers are used, which makes our conclusions without loss of generality. For training, we train all the networks using DIV2K training dataset (1). Our training program is implemented using the PyTorch framework (32) and is based on the wonderful BasicSR (37) toolbox. We use the Adam (20) optimization method with $\beta_1 = 0.9$, $\beta_2 = 0.999$. For learning rate, We select four fixed learning rates (1×10^{-3},1×10^{-4},5×10^{-5},2×10^{-5}) from a large range and an initial learning rate 2×10^{-4} with cosine decay function and warm restarts strategy (28). For pruning, we follow LTH's settings for deep neural networks, so we do not prune the first and last layers of the model and conduct a global pruning in all three methods. For LTH-PWC (27), we choose to fine-tune for another 1,000K iterations after 1,000K iterations' training. More detailed experimental settings are shown in Table 2.

Method	p	t	T
LTH	20%	0	1,000K / 300K / 100K
LMC-LTH	20%	60 / 18 / 6	1,000K / 300K / 100K
LTH-PWC	20%	1,000K	1,000K

Table 2: Experimental settings of three methods. p denotes the ratio of parameters to be pruned to the parameters that remains per pruning round. t denotes the iteration number of parameters loaded after pruning. T denotes the total training iterations.

A.2.2 Kernel level

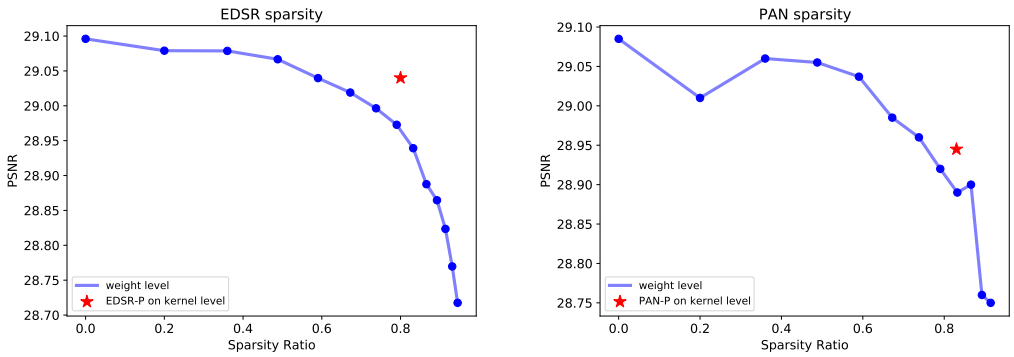
In this part, we also use EDSR (25) as the backbone large network. For grouped convolution, we use the official built-in implementation of PyTorch. For relational graph based convolutions, we follow the official implementation and integrate it into the SR networks. The randomly-connected convolutions are implemented with similar way as it can be viewed as a special case of relational graph based convolutions. For EDSR, we use the same settings as weight-level, iterations are set to 1000K. The only difference is that we only use the Cosine learning rate in weight-level part. For pruning part, we do not prune the first layer and the last layer of the network and we will apply 500K iterations for fine-tune with the same training strategy.

A.3 Width-sparsity on weight level

Figure 6a and Figure 6b show the results of finding trade-off on EDSR and pan with extended channels through "Lottery ticket hypothesis" (LTH). Because we want to find a convenient way to improve the performance of the model, we have not changed any training strategy during the pruning process. Specifically, all the experiments in the figure are conducted at the iteration number of 1000K, an initial learning rate 2×10^{-4} with cosine decay function and warm restarts strategy. We use the Adam optimization method with $\beta_1 = 0.9$, $\beta_2 = 0.999$.

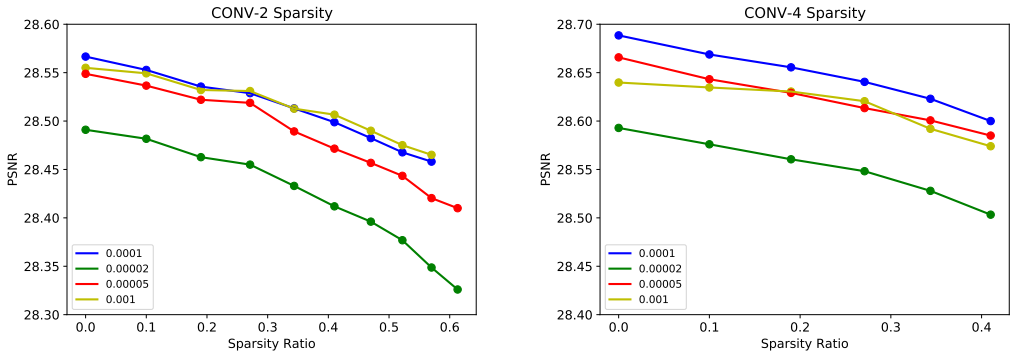
First, in Figure 6a, the performance of the network decreases with the increase of sparsity. At the same time, under the low sparsity, when the number of training iterations is 1,000K, the performance is not as good as the result of our sparsity on the kernel granularity.

In Figure 6b, the same phenomenon as EDSR appears. At the same time, when the network sparsity of PAN (47) is low, it is difficult for us to train the model to the end. The problem of model collapse will occur between 300K and 500K iterations. Therefore, in these sub-networks, we selected the PSNR maximum value that appeared in the training process. As the degree of sparsity increases, we can complete the training process. But the model effect is still not as good as the sparse model at the kernel level under the same number of channels and sparsity.



(a) (b)

Figure 6: (a) Experimental results of 128-channel EDSR. The x-axis represents the degree of network sparsity. The y-axis represents the PSNR tested on Div2k_val on data set. (b) Experimental results of 100-channel PAN. The x-axis represents the degree of network sparsity. The y-axis represents the PSNR tested on Div2k_val on data set.



(a) (b)

Figure 7: (a) Experimental results of Conv-2 toy model. The x-axis represents the degree of network sparsity. The y-axis represents the PSNR tested on Div2k_val on data set. (b) Experimental results of Conv-4 toy model. The x-axis represents the degree of network sparsity. The y-axis represents the PSNR tested on Div2k_val on data set.

In the experiments of LTH (10) , the higher the degree of network sparsity, the higher the number of iterations required to train the network. Therefore, thinning out on too fine-grained granularity will slow the convergence speed of the model. Because of this, sparsity at the kernel level will be more advantageous under the same number of iterations.

A.4 Toy models

We use 64 channel two-layer convolution and four-layer convolution to replace the 15 residual blocks in EDSR (25) . We selected four fixed learning rates (0.001, 0.0001, 0.00005, 0.00002), In order to sufficiently train the network, we still set the number of training iterations to 1000K. Meanwhile, we followed the experimental setup in LTH, so we set the p of each pruning to 10% and apply a layer-wise pruning (prune each layer separately at the same rate). Although we have not thinned out the two toy models to less than 10% sparsity, we can still observe the experimental phenomenon from Figure 7a and Figure 7b. That is, different from LTH in classification task, we could not find a sparse network that can achieve the expression ability of its fully connected network (In the LTH experiment, compared with deep neural networks, the “winning ticket" of a toy model is easier to find).

A.5 Quantitative results

Figure 8 and Table 3 shows the experimental results of our width-sparsity trade-off method. We are pleasantly surprised to find that our method can work not only on the midsize model Lattice with a parameters of 777k, but also on small network with only 270k parameters such as PAN. We divide the sparse network into performance and efficient modes (denoted "P" and "E" in Table 3). Efficient mode can reduce flops by 20% - 40% while maintaining the original performance of the network, performance mode can improve the PSNR on the Div2k_val set by 0.04dB-0.8dB while maintaining the FLOPS of the original network.

A.6 Qualitative results

Figure 9 shows the visualization effect of our proposed method. It can be seen that the proposed method improves the processing of dense texture in the network.

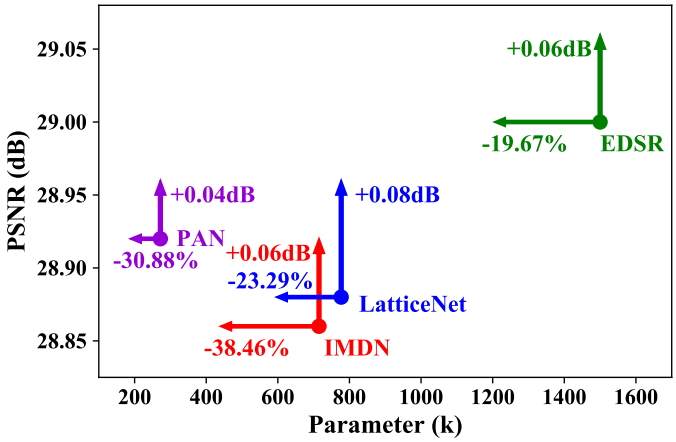


Figure 8: Improvement of PAN, IMDN, LatticeNet, and EDSR models through width-sparsity trade-off on kernel granularity. This is an extremely convenient method that can be deployed on most existing SR models. Parameter (x-axis) represents the number of parameters actually used by the network.

Model	Mode	channel	sparsity	Params(K)	BSD100		Urban100		DIV2K_val	
					psnr	ssim	psnr	ssim	psnr	ssim
EDSR	O	64	100%	1500	26.24	0.7118	24.60	0.7654	29.03	0.8205
	E	128	20%	1205	26.24	0.7123	24.59	0.7654	29.04	0.8206
	P	150	17%	1503	26.28	0.7133	24.70	0.7693	29.08	0.8217
PAN	O	40	100%	272	26.18	0.7108	24.44	0.7616	28.92	0.8173
	E	80	17%	188	26.17	0.7105	24.44	0.7614	28.91	0.8174
	P	80	25%	274	26.21	0.7108	24.51	0.7623	28.96	0.8182
IMDN	O	64	100%	1500	26.24	0.7118	24.60	0.7654	29.00	0.8195
	E	128	20%	1205	26.24	0.7123	24.58	0.7653	29.02	0.8202
	P	150	17%	1420	26.28	0.7133	24 70	0.7693	29.06	0.8213
LatticeNet	O	64	100%	777	26.17	0.7097	24.43	0.7594	28.88	0.8156
	E	128	17%	596	26.19	0.7101	24.45	0.7595	28.92	0.8172
	P	128	25%	780	26.22	0.7112	24.53	0.7628	28.96	0.8183

Table 3: Quantitative results of baseline convolutional networks and other sparse versions for scaling factor $\times 4$, "O" denotes the original network, "E" denotes the Efficiency-version, "P" denotes the Performance-version, and all experimental results of each model are from the same training settings.

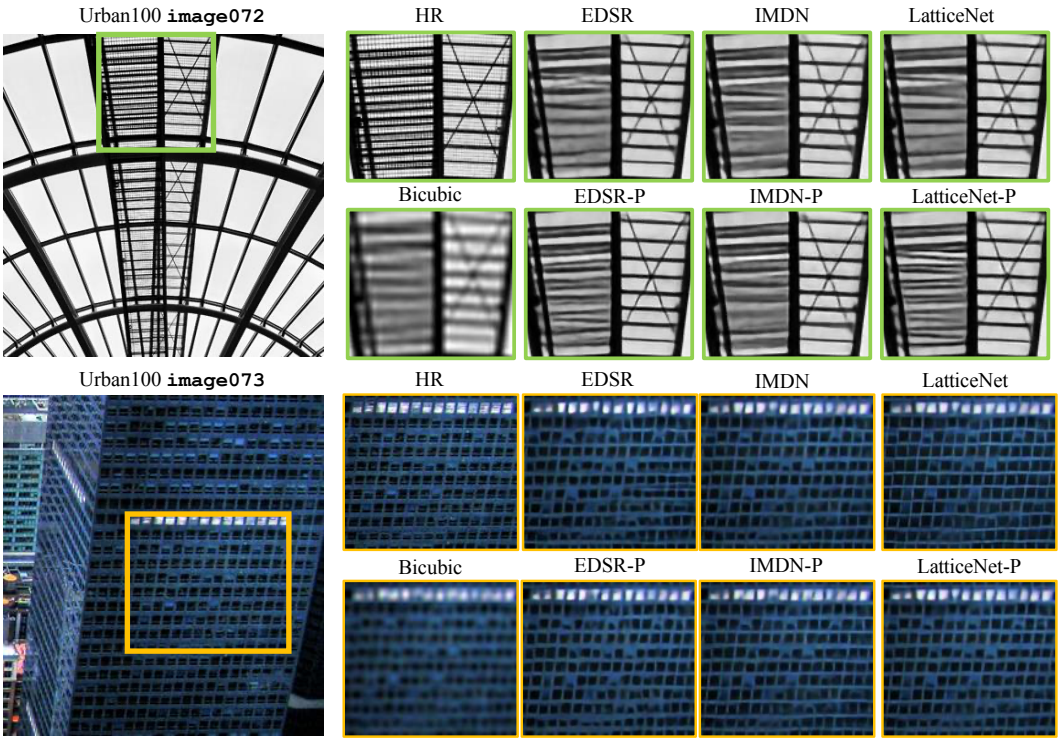


Figure 9: Qualitative results of our method on Urban100 data set. We use "- P" as the suffix to indicate the networks' performance mode.