LEARNING EFFICIENT IMAGE SUPER-RESOLUTION NETWORKS VIA STRUCTURE-REGULARIZED PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Lightweight image super-resolution (SR) networks have obtained promising results with moderate model size. However, they are impractical or neglected to be extended to larger networks. At the same time, model compression techniques, like neural architecture search and knowledge distillation, typically consume considerable computation resources. In contrast, network pruning is a cheap and effective model compression technique. However, it is hard to be applied to SR networks directly, because filter pruning for residual blocks is well-known tricky. To address the above issues, we propose structure-regularized pruning (SRP), which imposes regularization on the pruned structure to make sure the locations of pruned filters are aligned across different layers. Specifically, for the layers connected by the same residual, we select the filters of the same indices as unimportant filters. To transfer the expressive power in the unimportant filters to the rest of the network, we employ L_2 regularization to drive the weights towards zero so that eventually their absence will cause minimal performance degradation. We apply SRP to train efficient image SR networks, resulting in a lightweight network SRPN-L and a very deep one SRPN. We conduct extensive comparisons with both lightweight and larger image SR networks. SRPN-L and SRPN achieve superior performance gains over recent methods quantitatively and visually.

1 INTRODUCTION

Image super-resolution (SR) is a fundamental computer vision task, which aims to recover a high-resolution (HR) image from its low-resolution (LR) counterpart. In general, image SR is ill-posed as a many-to-one mapping problem. To alleviate this problem, plenty of deep convolutional neural networks (CNNs) (Dong et al., 2014; 2016; Kim et al., 2016b; Tai et al., 2017b) have been investigated to achieve accurate mapping from LR image to its HR target.

Deep CNN is firstly utilized for image SR in SRCNN (Dong et al., 2014) and continuously shows promising SR performance. There are only three convolutional (Conv) layers in SRCNN, hindering its performance. Kim *et al*. increased the network depth in VDSR (Kim et al., 2016a) with residual learning and obtained notable improvements over SRCNN. Lim *et al*. (Lim et al., 2017) built a much deeper network EDSR by using simplified residual blocks. Zhang *et al*. (Zhang et al., 2018b) proposed RCAN, which is one of the deepest networks in SR. With increased network size, very deep networks, like EDSR (Lim et al., 2017) and RCAN (Zhang et al., 2018b), have achieved remarkable SR performance. However, they also suffer from heavy model parameters, number of operations, and inference time. Therefore, it is infeasible to directly deploy them on resource-limited platforms without neural processing units or off-chip memory (Lee et al., 2020).

From this point of view, more and more works introduce efficient network architectures for lightweight SR networks (Ahn et al., 2018; Luo et al., 2020). Ahn *et al*. proposed cascading residual network (CARN) (Ahn et al., 2018). Hui *et al*. proposed information multi-distillation network (IMDN) (Hui et al., 2019). Lee *et al*. introduced knowledge distillation (Hinton et al., 2014) for image SR (Lee et al., 2020). Besides, neural architecture search (NAS) (Zoph & Le, 2017; Elsken et al., 2019) was also utilized for lightweight SR model (Chu et al., 2019a). However, there are still several downsides to these networks: **(1)** There is still an obvious performance gap between those lightweight models and the very deep models; **(2)** The training of these methods can consume heavy computation resources. For example, 8 Tesla V100 GPUs are needed to train a single network for

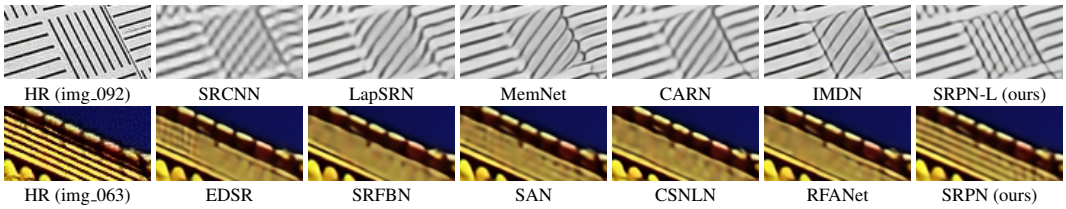

Figure 1: Visual comparisons ($\times$4) about lightweight and large SR networks on Urban100 dataset. The first row is lightweight SR networks comparison. The second row is about larger models comparison. SRPN-L and SRPN are our proposed lightweight and larger image SR networks.

about three days in (Chu et al., 2019a); **(3)** It is hard for most lightweight SR network designs to generalize to very large networks and achieve superior performance at the same time. In short, more effective, resource friendly, and generally lightweight SR networks are still in need.

On the other hand, neural network pruning is well-known as an effective technique to reduce model complexity (Reed, 1993; Sze et al., 2017). For acceleration, researchers mainly focus on filter pruning (a.k.a. structured pruning) (Li et al., 2017) rather than weight-element pruning (a.k.a. unstructured pruning) (Han et al., 2015; 2016b). Marrying filter pruning with image SR seems a plausible solution to strike a better performance-complexity trade-off. However, filter pruning methods in classification can hardly translate to SR networks directly. The main reason is that, residual connections are well-known hard to prune in structured pruning (Li et al., 2017) while they are *extensively* used in state-of-the-art SR networks (e.g., EDSR (Lim et al., 2017) has 32 residual blocks; RCAN (Zhang et al., 2018b) even has nested residual blocks).

To tackle the above issue, we propose *Structure-Regularized Pruning* (SRP), which imposes regularization on the pruned structure to make sure the locations of pruned filters are *aligned* across different layers. Specifically, for the layers connected by the same residual, we select the filters of the same indices as unimportant filters (i.e., those we will remove finally). In order to transfer the expressive power in the unimportant filters to the remainder of the network, we employ L_2 regularization to drive the weights towards zero gradually so that eventually their absence will incur negligible performance loss. To the best of our knowledge, our SRP is the first attempt to leverage structured pruning for efficient image SR. Our main contributions can be summarized as follows:

- We propose a network structure-regularized pruning (SRP) method to learn efficient image SR networks. We try to jointly train image SR models with network pruning simultaneously to achieve high reconstruction performance as well as efficiency.
- Our SRP provides a general idea to structurally prune networks with extensive residual connections. The introduction of regularization as a pruning tool manages to maintain the expressivity of the original network while peeling off the unnecessary redundancy.
- We employ SRP to train efficient image SR networks, resulting in a lightweight network (named SRPN-L) and a very deep one (named SRPN). We achieve superior performance gains on both lightweight and large image SR networks.

2 Related Work

Deep Image SR Models. Dong *et al*. (Dong et al., 2014) firstly proposed SRCNN for image SR and achieved superior performance with only 3 convolutional (Conv) layers. Residual learning was introduced in VDSR (Kim et al., 2016a), reaching 20 Conv layers and a significant improvement over SRCNN. Tai *et al*. later introduced memory block in MemNet (Tai et al., 2017b) for deeper network structure. Lim *et al*. (Lim et al., 2017) simplified the residual block (He et al., 2016) and constructed deeper and wider networks with a large number of parameters. Zhang *et al*. (Zhang et al., 2018b) proposed an even deeper network, residual channel attention network (RCAN), where the attention mechanism was firstly introduced in image SR. Liu *et al*. proposed FRANet (Liu et al., 2020) to make the residual features more focused on critical spatial contents. Later, Zhang *et al*. (Zhang et al., 2019) proposed residual non-local attention for image restoration, including image SR. Mei *et al*. proposed CSNLN (Mei et al., 2020) by combining local, in-scale/cross-scale non-local feature correlations, and external statistics. Most of them have achieved state-of-the-art results with deeper and wider networks. However, they suffer from huge model size (i.e., network parameter number) and/or heavy computation operations (i.e., FLOPs).

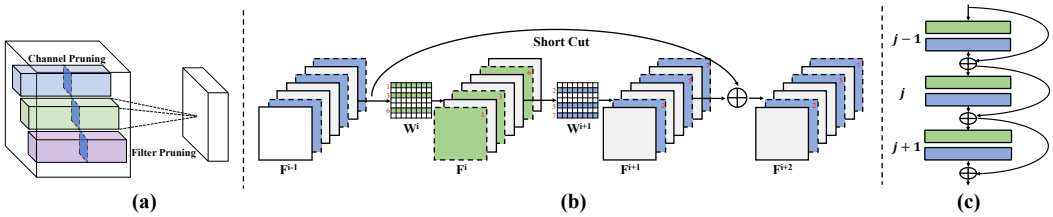

Figure 2: **(a)** Illustration of channel-wise and filter-wise pruning for single Conv layer. In this work, we adopt filter-wise pruning to learn efficient image SR networks. **(b)** Illustration of filter pruning *within* a residual block. Feature maps F are depicted as 3d cubes. Convolutional kernel W (4d tensor) is expended as a 2d matrix here for easy illustration (each row represents a filter). Both green and blue color mean the *pruned* filters (or feature map channels): green represents the pruned filters in *free Conv layers*; blue represents the pruned filters in *constrained Conv layers*. The main point of SRP is to apply L_2 regularization to the unimportant filters to make sure the pruned indices of $F^{(i)}$ are exactly the same as those of $F^{(i+2)}$. **(c)** Illustration of filter pruning across different residual blocks. All the layers that are directly followed by the add operators are constrained Conv layers (in blue). Others are free Conv layers (in green).

Lightweight Image SR Models. Recent years have been rising interest in investigating lightweight image SR models. These approaches try to design lightweight architectures, which mainly take advantage of recursive learning and channel splitting. Kim *et al*. firstly introduced recursive learning in DRCN to decrease model size (Kim et al., 2016b). Ahn *et al*. designed a cascading mechanism upon a residual network in CARN (Ahn et al., 2018). Hui *et al*. proposed a lightweight information multi-distillation network (IMDN) (Hui et al., 2019). Luo *et al*. designed the lattice block with butterfly structures (Luo et al., 2020). Recently, neural architecture search was introduced for image SR in FALSR (Chu et al., 2019a). Besides, model compression techniques, like knowledge distillation, have been investigated for image SR. He *et al*. proposed knowledge distillation based feature-affinity for efficient image SR (He et al., 2020). Lee *et al*. trained a teacher network to distill its knowledge to a student (Lee et al., 2020). Although those lightweight networks have achieved great progress, we still need to investigate deeper for more efficient image SR models.

Neural Network Pruning. Pruning aims to remove parameters in a neural network without compromising its performance seriously (Reed, 1993; Sze et al., 2017). It mainly falls into two groups: filter pruning (a.k.a. structured pruning)[1] and weight-element pruning (a.k.a. unstructured pruning). The former aims to remove weights by filters (i.e., 4-d tensors), while the latter removes weights by single elements (i.e., a scalar). Structured pruning results in regular sparsity after pruning. It does *not* demand any special hardware features to achieve considerable practical acceleration. In contrast, unstructured pruning leads to irregular sparsity. Leveraging the irregular sparsity for acceleration typically demands special software libraries, while past works have shown the practical speedup is very limited (Wen et al., 2016), unless using customized hardware platform (Han et al., 2016a). In this paper, we focus on *filter pruning* for easy acceleration. Most efforts in pruning (mainly in classification task) have been spent on finding a better pruning criterion to select unimportant weights (Reed, 1993; Sze et al., 2017). Magnitude-based (Han et al., 2015; 2016b; Li et al., 2017) is the most prevailing criterion, which we will also employ to develop our method in this paper. As far as we know, no work before has managed to apply filter pruning to compressing image SR networks. This paper is meant to fill the blank.

3 Proposed Method

We first give a brief view of the problem setting about deep CNN for image SR. We also observe that there exists heavy redundancy in the networks. To pursue more efficient image SR networks, we then propose structure-regularized pruning (SRP) method to compress them.

3.1 Deep CNN for Image SR

Image super-resolution (SR) aims to reconstruct a high-resolution (also known as super-resolved) image I_{SR} from its low-resolution (LR) image I_{LR}. This image SR process can be described as

$$I_{SR} = \mathcal{F}_{SR}(I_{LR}; \Theta), \tag{1}$$

[1]Filter pruning is actually a sub-concept within structured pruning, while they are often used interchangeably in the literature. In this paper, we stick to this terminology practice.

where $\mathcal{F}_{SR}(\cdot)$ is the deep image SR network and Θ denotes the network parameters. We also model the LR image I_{LR} from its HR counterpart as a degradation process

$$I_{LR} = \mathcal{F}_{\downarrow_s}(I_{HR}), \tag{2}$$

where $\mathcal{F}_{\downarrow_s}(\cdot)$ downscales the original ground truth I_{HR} with scaling factor s. The downscaling process may introduce additional noise, blurring, compression, and/or other unknown artifacts. Meanwhile, high-frequency information will be lost more or less. Image SR models try to recover high-frequency information as much as possible.

In this work, we focus on efficient neural networks with relatively fewer parameters and computation operations, but comparable or even higher performance.

3.2 STRUCTURE-REGULARIZED PRUNING (SRP)

Pruned Index Constraint. Filter pruning in residual networks is well-known tricky because the add operators in residual blocks demand the pruned filter indices must be aligned. Filter pruning within a residual block is shown in Fig. 2(b). A typical residual block (e.g., in EDSR (Lim et al., 2017), RCAN (Zhang et al., 2018b)) is made up of two convolutional layers. All the convolutional layers can be categorized into two groups based on their connection relationship among one another. One group comprises the layers that can be pruned *without any constraint*, dubbed *free Conv layers* in this work; the other comprises layers in which the filters must be pruned *at the same indices*, dubbed *constrained Conv layers*. For a concrete example, in Fig. 2(b), the layer $W^{(i)}$ is a free Conv layer and layer $W^{(i+1)}$ is a constrained Conv layer.

Owing to the pruned index constraint problem, many filter pruning methods in classification simply do *not* prune the last Conv layer in a residual blocks (Li et al., 2017; Wang et al., 2021), which can still deliver considerable speedup of practical interest. But, this naive solution cannot translate to the image SR networks. The fundamental reason is that image SR networks typically employ *much more* residual blocks and each block typically has only two Conv layers. In some top-performed SR networks (e.g., RCAN), there are even *nested* residual blocks. To see how serious this problem is, taking EDSR as an example, it has 32 residual blocks and each block has two Conv layers. If we do not prune the 2nd Conv layer in a residual block, it means half of the Conv layers are not pruned. Namely, *at best*, we can only achieve $2\times$ theoretical acceleration (measured by FLOPs reduction).

Given the issue above, it is necessary to prune *all* the Conv layers in residual blocks if we seek acceleration of practical use. Thus, we need a method to align the pruned indices in all constrained Conv layers. Regularization then arises as a natural solution given its prevailing use to impose priors on the sparsity structure in pruning (Reed, 1993; Wen et al., 2016; Wang et al., 2021).

Our SRP method is based on regularization to induce desired sparsity structure. Primarily, there are three questions to answer in SRP: (1) which pruning criterion is used to choose unimportant filters; (2) which kind of regularization is chosen to induce sparsity; (3) how to schedule the pruning process to make the algorithm robust and easy to use.

(1) Pruning Criterion. When it comes to formulated pruning criteria, they mainly fall into three groups: magnitude based (Han et al., 2015; Li et al., 2017), 1st-order gradient based (Molchanov et al., 2017; 2019), and 2nd-order gradient based (LeCun et al., 1990; Hassibi & Stork, 1993; Wang et al., 2019a). The 1st-order and 2nd-order gradient based criteria are based on Taylor approximation of the increased loss when a weight is removed, pioneered by (LeCun et al., 1990). They are more accurate than magnitude based one right after pruning. However, pruning is typically followed by a finetuning process (Reed, 1993) to regain performance. After finetuning, the advantage of the 1st-order and 2nd-order gradient based criteria usually becomes marginal. Meanwhile, they are typically more costly than the magnitude based criteria. Taking the cost and flexibility into account, we simply employ magnitude based pruning criteria.

Specifically, given a pretrained SR network, for all the free Conv layers, we sort the filters in ascending order by their L_1-norms and select the filers with the least norms as *unimportant filters*, by a predefined pruning ratio $r \in (0, 1)$. For the constrained Conv layers, the problem becomes a little bit more complex. Given the pruned index constraint, ideally, we can only prune the filters in the *intersection* set (denoted by $\mathcal{S}^{(net)}$) for the whole network,

$$\mathcal{S}^{(net)} = \bigcap_{l \in \mathcal{C}} S^{(l)}, \tag{3}$$

where $\mathcal{S}^{(l)}$ represents the set of unimportant filters selected by certain pruning criterion in the l-th Conv layer; $\mathcal{C}$ denotes the set of constrained Conv layers. For top-performed image SR networks, $|\mathcal{C}|$ ($|\cdot|$ denotes cardinality of a set) is at the order of magnitude of hundreds (e.g., $|\mathcal{C}| = 200$ for RCAN). Meanwhile, given a pretrained model, its pruned filter indices by L_1-norm are usually *random* across different layers. As a consequence, $|\mathcal{S}^{(net)}|$ will be rather small (in practice, on RCAN, we even observe $|\mathcal{S}^{(net)}| = 0$ when we try to prune 32 filters out of 64). This issue brings much trouble to user control when we want to prune a layer to a pre-defined number of filters. To resolve this problem, we choose to *randomly* select a set of filters as unimportant and fixed during the whole pruning process. In practice, we find this simple scheme works pretty well. The reason enabling us to do so is that regularization for pruning is a *gradual* process. It allows abundant time for the network to recover from the introduced penalty term (this will be empirically verified in our experiments, see Fig. 3). Apart from aligning cross-layer sparsity structure, this is the second reason that we choose regularization for pruning in our algorithm.

(2) Regularization Form. Although L_1 regularization is well-known to induce sparsity in machine learning, it is hard to tune the coefficient to strike a good balance between sparsity and performance. Therefore, L_2 regularization is adopted in our method given its easy control. Specifically, given the loss function $\mathcal{L}$ of a neural network Θ, the total error function $\mathcal{E}$ with the proposed adaptive L_2 regularization term can be formulated as

$$\mathcal{E}(\Theta; \mathcal{D}) = \mathcal{L}(\Theta; \mathcal{D}) + \frac{1}{2} \sum_{l,j} \alpha_j^{(l)} ||W_j^{(l)}||_2^2, \tag{4}$$

where $\mathcal{D}$ stands for the training dataset; $W_j^{(l)}$ refers to the j-th filter in l-th Conv layer; $\alpha_j^{(l)}$ stands for the L_2 regularization co-efficient for that filter. If a filter j falls into the set of unimportant filters $S^{(l)}$, $\alpha_j^{(l)} > 0$; otherwise, $\alpha_j^{(l)} = 0$ since we do not restrict the learning of important filters.

(3) Regularization Schedule. As mentioned above, for the constrained Conv layers, we select the unimportant filters randomly. To mitigate the side effect of this sub-optimal choice, the network should be provided with abundant training iterations to adapt so that it can transfer its expressive power to the remaining part of the network. To achieve this goal, we choose to gradually increase the penalty strength, drawing inspirations from (Wang et al., 2019b; 2021),

$$\alpha_j^{(l)} = \alpha_j^{(l)} + \Delta, \tag{5}$$

where Δ (a pre-defined constant) is a small amount that the penalty strength is increased each time. This design allows us to raise the penalty strength to a very large amount eventually (like 0.5 in our experiments, note the normal L_2 regularization co-efficient is at the scale of 10^{-4}). As a result, the unimportant weights are driven very close to zero and removing them will hardly hurt the performance. Besides, α is updated every T iterations during training for stability.

As a termination condition, we introduce a ceiling limit τ (a pre-defined constant) for the regularization co-efficient α. When all the α for unimportant filters reaches τ, the pruning process is finished.

3.3 ARM IMAGE SR MODELS WITH SRP

The proposed SRP method can be applied to state-of-the-art SR methods in a *plug-and-play* manner – simply replace the loss function $\mathcal{L}$ in Eq. (4) with the loss function of an SR method. All the features in the original SR method can stay as they are. The proposed L_2 regularization can be implemented on any deep learning framework very easily.

When the pruning process is finished, we remove the unimportant filters (not zeroing out them only, but literally removing them from the model), which results in a small model. Then we finetune the small model to regain performance following the common practice in pruning (Reed, 1993).

3.4 IMPLEMENTATION DETAILS

Here we elaborate how to apply SRP to constructing lightweight and very deep image SR networks.

Lightweight Network. First, we revise EDSR baseline (i.e.16 residual blocks) (Lim et al., 2017) by removing the final Conv layer to save parameters. Same as IMDN (Hui et al., 2019), the reconstruction was done within the pixel-shuffle layer (Shi et al., 2016). We set the channel number in revised EDSR baseline as 256 and then prune it to 45. For $\times 2$, we reduce the number of parameters from 19.5M to 609K and name the compressed model as SRPN-L.

Pruning ratio	Params (K)	FLOPs (G)	Scratch	L_1-norm pruning (Li et al., 2017)	SRPN (ours)
0.1	1,101.8	254.5	37.85	37.91	**37.97**
0.3	681.1	157.7	37.81	37.81	**37.89**
0.5	381.8	88.9	37.75	37.73	**37.84**
0.7	154.2	36.5	37.56	37.58	**37.71**
0.9	26.9	7.3	36.74	36.87	**37.28**

Table 1: PSNR (dB) comparison on Set5 ($\times$2) between SRP and other two methods to obtain the same small network. "Scratch" stands for training the small network from scratch. "L_1-norm" removes filters with the smallest L_1-norms. The unpruned model is EDSR baseline (Params: 1,369.9K, FLOPs: 316.3G, PSNR: 37.99 dB). The best results are **highlighted**.

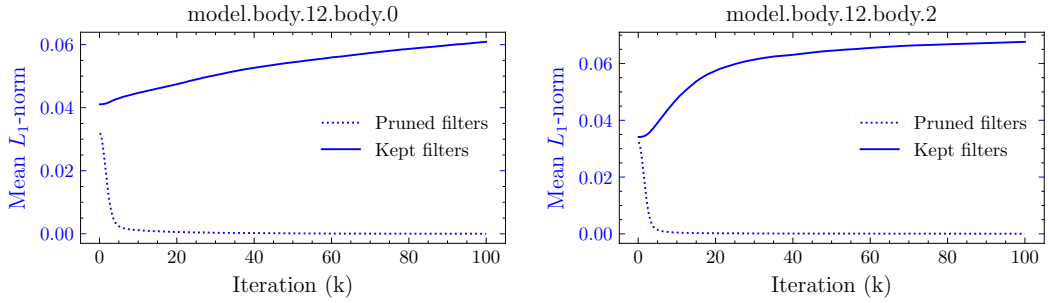

Figure 3: Illustration of the pruning process for "model.body.12.body.0" (free Conv layer) and "model.body.12.body.2" (constrained Conv layer) in EDSR. The mean L_1-norm of pruned or kept filters are plotted across different training iterations.

Very Deep Network. To apply SRP to very deep networks, like over 400 Conv layers in RCAN, we revise RCAN (Zhang et al., 2018b) by removing all channel attention modules. We set the channel number in the revised RCAN as 96 and then prune it to 64. For $\times$2, we reduce the number of parameters from 34.5M to 15.3M and name the compressed model as SRPN.

4 Experimental Results

4.1 Settings

Data and Evaluation. We use DIV2K dataset (Timofte et al., 2017) as training data, following most recent works (Timofte et al., 2017; Lim et al., 2017; Zhang et al., 2018a; Haris et al., 2018). For testing, we use five standard benchmark datasets: Set5 (Bevilacqua et al., 2012), Set14 (Zeyde et al., 2010), B100 (Martin et al., 2001), Urban100 (Huang et al., 2015), and Manga109 (Matsui et al., 2017). The SR results are evaluated with PSNR and SSIM (Wang et al., 2004) on Y channel of transformed YCbCr space. We also provide model size and FLOPs (a.k.a. Mult-Adds) comparisons.

Training Settings. Following (Zhang et al., 2018b), we perform data augmentation on the training images, which are randomly rotated by 90°, 180°, 270° and flipped horizontally. Each training batch consists of 16 LR color patches, whose size is 48$\times$48. Our SRP model is trained by ADAM optimizer (Kingma & Ba, 2014) with β_1=0.9, β_2=0.999, and ϵ=10^{-8}. We set the initial learning rate as 10^{-4} and then decrease it to half every 2×10^5 iterations of back-propagation. We use PyTorch (Paszke et al., 2017) to implement our models with a Tesla V100 GPU[2].

4.2 Ablation Study

For ablation study, we use EDSR baseline (i.e., 16 residual blocks) (Lim et al., 2017) as backbone, because it is a widely used image SR baseline with public code[3] and results.

Comparison with Baseline Methods. We first conduct ablation study to show the proposed pruning method is effective. Two baseline methods are compared: training from scratch and the original L_1-norm pruning (Li et al., 2017). The results are shown in Tab. 1. As seen, our pruned network delivers the best PSNR *consistently* across different pruning ratios. This shows our method is not

[2]We have attached a demo code in the supplementary file. All the code and trained models of SRP will be available after the paper is published.

[3]https://github.com/sanghyun-son/EDSR-PyTorch

Method	Scale	Params	Mult-Adds	Set5		Set14		B100		Urban100	
				PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
SRCNN	×2	57K	52.7G	36.66	0.9542	32.42	0.9063	31.36	0.8879	29.50	0.8946
FSRCNN	×2	12K	6.0G	37.00	0.9558	32.63	0.9088	31.53	0.8920	29.88	0.9020
VDSR	×2	665K	612.6G	37.53	0.9587	33.03	0.9124	31.90	0.8960	30.76	0.9140
DRCN	×2	1,774K	17,974.3G	37.63	0.9588	33.04	0.9118	31.85	0.8942	30.75	0.9133
LapSRN	×2	813K	29.9G	37.52	0.9590	33.08	0.9130	31.80	0.8950	30.41	0.9100
DRRN	×2	297K	6,796.9G	37.74	0.9591	33.23	0.9136	32.05	0.8973	31.23	0.9188
MemNet	×2	677K	2,662.4G	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195
CARN	×2	1,592K	222.8G	37.76	0.9590	33.52	0.9166	32.09	0.8978	31.92	0.9256
IMDN	×2	694K	158.8G	38.00	0.9605	33.63	0.9177	32.19	0.8996	32.17	0.9283
SRPN-L (ours)	×2	609K	139.9G	**38.10**	**0.9608**	**33.70**	**0.9189**	**32.25**	**0.9005**	**32.26**	**0.9294**
SRCNN	×3	57K	52.7G	32.75	0.9090	29.28	0.8209	28.41	0.7863	26.24	0.7989
FSRCNN	×3	12K	5.0G	33.16	0.9140	29.43	0.8242	28.53	0.7910	26.43	0.8080
VDSR	×3	665K	612.6G	33.66	0.9213	29.77	0.8314	28.82	0.7976	27.14	0.8279
DRCN	×3	1,774K	17,974.3G	33.82	0.9226	29.76	0.8311	28.80	0.7963	27.15	0.8276
DRRN	×3	297K	6,796.9G	34.03	0.9244	29.96	0.8349	28.95	0.8004	27.53	0.8378
MemNet	×3	677K	2,662.4G	34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376
CARN	×3	1,592K	118.8G	34.29	0.9255	30.29	0.8407	29.06	0.8034	28.06	0.8493
IMDN	×3	703K	71.5G	34.36	0.9270	30.32	0.8417	29.09	0.8046	28.17	0.8519
SRPN-L (ours)	×3	615K	62.7G	**34.47**	**0.9276**	**30.38**	**0.8425**	**29.16**	**0.8061**	**28.22**	**0.8534**
SRCNN	×4	57K	52.7G	30.48	0.8628	27.49	0.7503	26.90	0.7101	24.52	0.7221
FSRCNN	×4	12K	4.6G	30.71	0.8657	27.59	0.7535	26.98	0.7150	24.62	0.7280
VDSR	×4	665K	612.6G	31.35	0.8838	28.01	0.7674	27.29	0.7251	25.18	0.7524
DRCN	×4	1,774K	17,974.3G	31.53	0.8854	28.02	0.7670	27.23	0.7233	25.14	0.7510
LapSRN	×4	813K	149.4G	31.54	0.8850	28.19	0.7720	27.32	0.7280	25.21	0.7560
DRRN	×4	297K	6,796.9G	31.68	0.8888	28.21	0.7720	27.38	0.7284	25.44	0.7638
MemNet	×4	677K	2,662.4G	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630
CARN	×4	1,592K	90.9G	32.13	0.8937	28.60	0.7806	27.58	0.7349	26.07	0.7837
IMDN	×4	715K	40.9G	32.21	0.8948	28.58	0.7811	27.56	0.7353	26.04	0.7838
SRPN-L (ours)	×4	623K	35.8G	**32.24**	**0.8958**	**28.69**	**0.7836**	**27.63**	**0.7373**	**26.16**	**0.7875**

Table 2: Quantitative results of lightweight SR networks. Best results are **highlighted**.

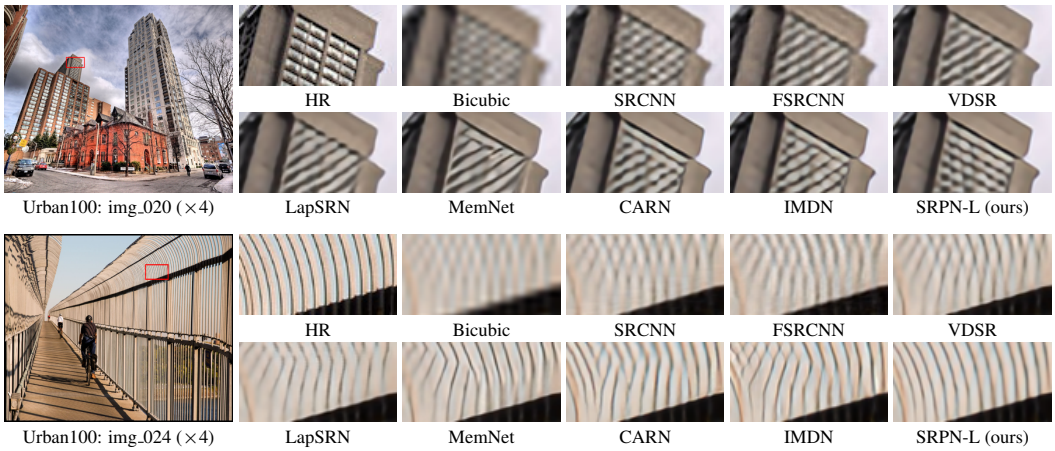

Figure 4: Visual comparison (×4) with lightweight SR networks on Urban100 dataset.

only effective (better than scratch training), but also more effective than naively applying pruning method in classification to SR (better than L_1-norm). In addition, under a *larger* pruning ratio, the advantage of our method over scratch training or L_1-norm is *more pronounced* in general, suggesting that our method is more valuable in the extreme pruning cases. Notably, our method also adopts the L_1-norm as pruning criterion, the same as (Li et al., 2017). However, our results are significantly better than theirs. The reason is that they do not impose any regularization on the pruned structure, thus the kept feature map channels are *misaligned* in residual blocks after pruning. In contrast, our method does not have this problem, thanks to the proposed structure regularization.

Visualization of Pruning Process. To figuratively understand how SRP works, in Fig. 3 we plot the mean L_1-norm of filters in two layers of EDSR baseline during SRP training. The filters are split into two groups, pruned and kept. As seen, the mean L_1-norm of the pruned filters goes down gradually because the penalty grows stronger and stronger, driving them towards zero. Interestingly, note the L_1-norms of the kept filters arise themselves (we do not have any regularization term to encourage them to grow larger). It means the network *learns to recover itself*, akin to the compensation effect in human brain (Duffau et al., 2003). We provide more visualization results in the appendix.

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
EDSR	×2	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
DBPN	×2	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
RCAN	×2	38.27	0.9614	34.12	0.9216	32.41	0.9027	33.34	0.9384	39.44	0.9786
SRFBN	×2	38.11	0.9609	33.82	0.9196	32.29	0.9010	32.62	0.9328	39.08	0.9779
SAN	×2	38.31	**0.9620**	34.07	0.9213	32.42	0.9028	33.10	0.9370	39.32	0.9792
HAN	×2	38.27	0.9614	34.16	0.9217	32.41	0.9027	33.35	0.9385	39.46	0.9785
IGNN	×2	38.24	0.9613	34.07	0.9217	32.41	0.9025	33.23	0.9383	39.35	0.9786
CSNLN	×2	38.28	0.9616	34.12	0.9223	32.40	0.9024	33.25	0.9386	39.37	0.9785
RFANet	×2	38.26	0.9615	34.16	0.9220	32.41	0.9026	33.33	0.9389	39.44	0.9783
SRPN (ours)	×2	**38.34**	0.9619	**34.29**	**0.9232**	**32.47**	**0.9032**	**33.50**	**0.9401**	**39.76**	**0.9796**
EDSR	×3	34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653	34.17	0.9476
RCAN	×3	34.74	0.9299	30.65	0.8482	29.32	0.8111	29.09	0.8702	34.44	0.9499
SRFBN	×3	34.70	0.9292	30.51	0.8461	29.24	0.8084	28.73	0.8641	34.18	0.9481
SAN	×3	34.75	0.9300	30.59	0.8476	29.33	0.8112	28.93	0.8671	34.30	0.9494
HAN	×3	34.75	0.9299	30.67	0.8483	29.32	0.8110	29.10	0.8705	34.48	0.9500
IGNN	×3	34.72	0.9298	30.66	0.8484	29.31	0.8105	29.03	0.8696	34.39	0.9496
CSNLN	×3	34.74	0.9300	30.66	0.8482	29.33	0.8105	29.13	0.8712	34.45	0.9502
RFANet	×3	34.79	0.9300	30.67	0.8487	29.34	0.8115	29.15	0.8720	34.59	0.9506
SRPN (ours)	×3	**34.84**	**0.9303**	**30.76**	**0.8497**	**29.39**	**0.8120**	**29.36**	**0.8749**	**34.88**	**0.9515**
EDSR	×4	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.02	0.9148
SRMDNF	×4	31.96	0.8925	28.35	0.7787	27.49	0.7337	25.68	0.7731	30.09	0.9024
RCAN	×4	32.63	0.9002	28.87	0.7889	27.77	0.7436	26.82	0.8087	31.22	0.9173
SRFBN	×4	32.47	0.8983	28.81	0.7868	27.72	0.7409	26.60	0.8015	31.15	0.9160
SAN	×4	32.64	0.9003	28.92	0.7888	27.78	0.7436	26.79	0.8068	31.18	0.9169
HAN	×4	32.64	0.9002	28.90	0.7890	27.80	0.7442	26.85	0.8094	31.42	0.9177
IGNN	×4	32.57	0.8998	28.85	0.7891	27.77	0.7434	26.84	0.8090	31.28	0.9182
CSNLN	×4	32.68	0.9004	28.95	0.7888	27.80	0.7439	27.22	0.8168	31.43	0.9201
RFANet	×4	32.66	0.9004	28.88	0.7894	27.79	0.7442	26.92	0.8112	31.41	0.9187
SRPN (ours)	×4	**32.72**	**0.9010**	**29.04**	**0.7917**	**27.87**	**0.7457**	**27.25**	**0.8172**	**31.82**	**0.9221**

Table 3: Quantitative results of large SR networks. Best results are **highlighted**.

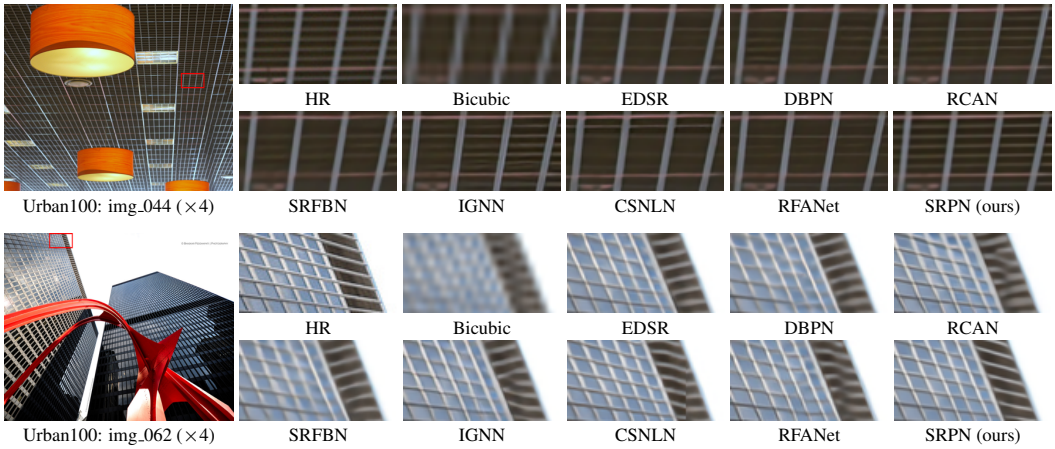

Figure 5: Visual comparison (×4) with large SR networks on Urban100 dataset.

4.3 Comparisons with Lightweight Networks

We compare our SRPN-L with lightweight SR networks: SRCNN (Dong et al., 2014), FSRCNN (Dong et al., 2016), VDSR (Kim et al., 2016a), DRCN (Kim et al., 2016b), LapSRN (Lai et al., 2017), DRRN (Tai et al., 2017a), MemNet (Tai et al., 2017b), CARN (Ahn et al., 2018) and IMDN (Hui et al., 2019). We provide more comparisons (e.g., PAN (Zhao et al., 2020)) in appendix.

Performance Comparisons. Table 2 shows PSNR/SSIM comparisons for ×2, ×3, and ×4 SR. When compared with all previous methods, our SRPN-L performs the best on all the datasets with all scaling factors. Different from careful network designs as most compared methods have done, we start with the existing EDSR baseline (Lim et al., 2017) and prune it to a much smaller network, showing the effectiveness of our proposed SRP.

Model Size and Mult-Adds. Compared with recent works (e.g., MemNet, CARN, and IMDN), our SRPN-L has the least parameter number. We also provide operations number with Mult-Adds by setting the output size as 3×1280×720. Our SRPN-L operates less Mult-Adds than most compared methods. Those comparisons indicate that SRP reduces parameters and operations efficiently.

Method	Params	Mult-Adds	Set5	Set14	B100	Urban100
MoreMNAS-A (Chu et al., 2019b)	1,039K	238.6G	37.63	33.23	31.95	31.24
FALSR-A (Chu et al., 2019a)	1,021K	234.7G	37.82	33.55	32.12	31.93
CARN+KD (Lee et al., 2020)	1,592K	222.8G	37.82	N/A	32.08	N/A
SRPN-L (ours)	609K	139.9G	38.10	33.70	32.25	32.26

Table 4: Model complexity comparisons ($\times 2$). Output size is $3 \times 1280 \times 720$ to calculate Mult-Adds.

Visual Comparisons. We further provide visual comparisons ($\times 4$) in Fig. 4 for challenging cases. We can observe that most of the compared methods cannot recover structural details with proper directions and suffer from blurring artifacts. In contrast, our SRPN-L can better alleviate the blurring artifacts and recover more structures. These visual comparisons are consistent with the quantitative results, demonstrating the superiority of our SRP method.

4.4 Comparisons with Larger Networks

Most previous lightweight SR networks neglect to extend their models to deeper and compare with larger networks. As stated in Sec. 3.4, we extend our SRP method to larger network training and obtain a deeper network SRPN. In Tab. 3, we compare with large SR networks: EDSR (Lim et al., 2017), DBPN (Haris et al., 2018), RCAN (Zhang et al., 2018b), SRFBN (Li et al., 2019), SAN (Dai et al., 2019), HAN (Niu et al., 2020), IGNN (Zhou et al., 2020), CSNLN (Mei et al., 2020), and RFANet (Liu et al., 2020). We provide more comparisons (e.g., DRN (Guo et al., 2020)) in appendix.

Performance Comparisons. In Tab. 3, we observe that our SRPN achieves the best PSNR/SSIM values across different test sets and scales, except for SSIM value on Set5 ($\times 2$). Our SRPN utilizes slightly fewer parameters (15.3M vs. 15.4M) with same network depth as RCAN (Zhang et al., 2018b). Notably, our SRPN removes the channel attention module in RCAN, but still achieves better results. This is mainly because our SRPN is initially trained from a wider and stronger network (with 96 channels). We train the much larger network and then prune it to the target size.

Visual Comparisons. We further show visual comparisons with larger networks (e.g., CSNLN, RFANet, and IGNN) in Fig. 5, we can also observe similar situations as in Fig. 4. For example, in img_011, other methods suffer from heavy blurring artifacts. In other cases, they may not recover clear structures in img_044 or even wrong details in img_062. Our SRPN achieves more visually pleasing results with more structural details. Those comparisons demonstrate that our SRP method can be applied to train larger networks and obtain comparable or even better performance.

4.5 Model Compression Comparisons

We further compare our network pruning method with representative model compression techniques for image SR. Specifically, we compare with neural architecture search based methods (Chu et al., 2019b;a) and knowledge distillation (KD) based methods (Lee et al., 2020). We provide quantitative results in Tab. 4. Our SRPN-L obtains the best performance with the least parameter number and Mult-Adds. With our SRP pruning method, we do not have to search lots of architectures or train a teacher network, which consumes extra computation resources. These comparisons show that our SRP method has great potential for efficient image SR. We provide more discussions and comparisons with related works (e.g., DCP (Zhuang et al., 2018), DHP (Li et al., 2020)) in the appendix.

5 Conclusion

Lately, researchers have investigated lightweight image super-resolution (SR) networks and achieved promising results with moderate model size and FLOPs. However, they neglect to extend their models to larger networks, which usually perform much better with more parameters and/or operations with residual learning. Meanwhile, model compression techniques, like neural architecture search and knowledge distillation, have also been introduced for efficient SR networks. However, they usually consume expensive computation resources. Network pruning, as another promising model compression method, is hard to train lightweight SR networks directly. To address these issues, we propose structure-regularized pruning (SRP), which imposes L_2 regularization on the pruned structure to align the locations of pruned filters across different layers. It can transfer the expressive power in the unimportant filters to the remaining part of the network. We employ SRP to train both lightweight and large image SR networks, resulting in a lightweight network SRPN-L and a very deep one SRPN. Our SRPN-L and SRPN achieve superior performance over recent state-of-the-art methods quantitatively and qualitatively.

REFERENCES

Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, 2018. 1, 3, 8, 14, 15

Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012. 6

Jae-Seok Choi and Munchurl Kim. A deep convolutional neural network with selection units for super-resolution. In *CVPRW*, 2017. 15

Jun-Ho Choi, Jun-Hyuk Kim, Manri Cheon, and Jong-Seok Lee. Lightweight and efficient image super-resolution with block state-based recursive network. *arXiv preprint arXiv:1811.12546*, 2018. 15

Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. *arXiv preprint arXiv:1901.07261*, 2019a. 1, 2, 3, 9, 15

Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Hailong Ma. Multi-objective reinforced evolution in mobile neural architecture search. *arXiv preprint arXiv:1901.01074*, 2019b. 9

Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *CVPR*, 2019. 9, 16

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 1, 2, 8, 15

Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016. 1, 8, 15

H Duffau, L Capelle, D Denvil, N Sichez, P Gatignol, M Lopes, MC Mitchell, JP Sichez, and R Van Effenterre. Functional recovery after surgical resection of low grade gliomas in eloquent brain: hypothesis of brain compensation. *Journal of Neurology, Neurosurgery & Psychiatry*, 74 (7):901–907, 2003. 7

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *JMLR*, 2019. 1

Yuchen Fan, Honghui Shi, Jiahui Yu, Ding Liu, Wei Han, Haichao Yu, Zhangyang Wang, Xinchao Wang, and Thomas S Huang. Balanced two-stage residual networks for image super-resolution. In *CVPRW*, 2017. 15

Yuchen Fan, Jiahui Yu, Yiqun Mei, Yulun Zhang, Yun Fu, Ding Liu, and Thomas S Huang. Neural sparse representation for image restoration. In *NeurIPS*, 2020. 16

Yong Guo, Jian Chen, Jingdong Wang, Qi Chen, Jiezhang Cao, Zeshuai Deng, Yanwu Xu, and Mingkui Tan. Closed-loop matters: Dual regression networks for single image super-resolution. In *CVPR*, 2020. 9, 14

S. Han, X. Liu, and H. Mao. EIE: efficient inference engine on compressed deep neural network. *ACM Sigarch Computer Architecture News*, 44(3):243–254, 2016a. 3

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015. 2, 3, 4

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016b. 2, 3

Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *CVPR*, 2018. 6, 9, 15

B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NeurIPS*, 1993. 4

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

Xiangyu He, Zitao Mo, Peisong Wang, Yang Liu, Mingyuan Yang, and Jian Cheng. Ode-inspired network design for single image super-resolution. In *CVPR*, 2019. 16

Zibin He, Tao Dai, Jian Lu, Yong Jiang, and Shu-Tao Xia. Fakd: Feature-affinity based knowledge distillation for efficient image super-resolution. In *ICIP*, 2020. 3

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Workshop*, 2014. 1

Zejiang Hou and Sun-Yuan Kung. Efficient image super resolution via channel discriminative deep neural network pruning. In *ICASSP*, 2020. 15

Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 6

Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACM MM*, 2019. 1, 3, 5, 8, 14, 15

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016a. 1, 2, 8, 15

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016b. 1, 3, 8, 15

Jun-Hyuk Kim, Jun-Ho Choi, Jaehyuk Chang, and Jong-Seok Lee. Efficient deep learning-based lossy image compression via asymmetric autoencoder and pruning. In *ICASSP*, 2020. 15

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 6

Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 8, 15

Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *NeurIPS*, 1990. 4

Wonkyung Lee, Junghyup Lee, Dohyung Kim, and Bumsub Ham. Learning with privileged information for efficient image super-resolution. In *ECCV*, 2020. 1, 3, 9

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017. 2, 3, 4, 6, 7

Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks. In *ECCV*, 2020. 9, 14

Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu. Feedback network for image super-resolution. In *CVPR*, 2019. 9, 16

Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017. 1, 2, 4, 5, 6, 8, 9, 15

Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. Non-local recurrent network for image restoration. In *NeurIPS*, 2018. 15

Jie Liu, Wenjie Zhang, Yuting Tang, Jie Tang, and Gangshan Wu. Residual feature aggregation network for image super-resolution. In *CVPR*, 2020. 2, 9, 16

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017. 14

Xiaotong Luo, Yuan Xie, Yulun Zhang, Yanyun Qu, Cuihua Li, and Yun Fu. Latticenet: Towards lightweight image super-resolution with lattice block. In *ECCV*, 2020. 1, 3, 14, 15

David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 6

Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 2017. 6

Yiqun Mei, Yuchen Fan, Yuqian Zhou, Lichao Huang, Thomas S Huang, and Humphrey Shi. Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining. In *CVPR*, 2020. 2, 9, 16

P. Molchanov, S. Tyree, and T. Karras. Pruning convolutional neural networks for resource efficient inference. In *ICLR*, 2017. 4

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *CVPR*, 2019. 4

Ben Niu, Weilei Wen, Wenqi Ren, Xiangde Zhang, Lianping Yang, Shuzhen Wang, Kaihao Zhang, Xiaochun Cao, and Haifeng Shen. Single image super-resolution via a holistic attention network. In *ECCV*, 2020. 9, 16

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

R. Reed. Pruning algorithms – a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, 1993. 2, 3, 4, 5

Haoyu Ren, Mostafa El-Khamy, and Jungwon Lee. Image super resolution based on fusing multiple convolution neural networks. In *CVPRW*, 2017. 15

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 *CVPR*, 2016. 5

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017. 2, 3

Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017a. 8, 15

Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, 2017b. 1, 2, 8, 15

Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, 2017. 6

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *ICML*, 2019a. 4

Huan Wang, Xinyi Hu, Qiming Zhang, Yuehai Wang, Lu Yu, and Haoji Hu. Structured pruning for efficient convolutional neural networks via incremental regularization. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):775–788, 2019b. 5

Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. In *ICLR*, 2021. 4, 5

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 6

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NeurIPS*, 2016. 3, 4

Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Proc. 7th Int. Conf. Curves Surf.*, 2010. 6

Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR*, 2018a. 6, 15

Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018b. 1, 2, 4, 6, 9, 15

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018c. 15

Yulun Zhang, Kunpeng Li, Kai Li, Bineng Zhong, and Yun Fu. Residual non-local attention networks for image restoration. In *ICLR*, 2019. 2, 15

Hengyuan Zhao, Xiangtao Kong, Jingwen He, Yu Qiao, and Chao Dong. Efficient image super-resolution using pixel attention. In *ECCVW*, 2020. 8, 14

Shangchen Zhou, Jiawei Zhang, Wangmeng Zuo, and Chen Change Loy. Cross-scale internal graph neural network for image super-resolution. In *NeurIPS*, 2020. 9, 16

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *NeurIPS*, 2018. 9, 14

Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 1

A MORE DISCUSSIONS AND INVESTIGATIONS

A.1 DIFFERENCES WITH DHP

(1) The main idea of DHP (Li et al., 2020) is to employ meta-learning to automatically decide *different* layer-wise pruning ratios, while SRP simply employs the *same* pre-defined ratio for different layers. (2) DHP (Li et al., 2020) *learns* unimportant filters via regularization, while our SRP picks them via a pre-defined importance criterion (i.e., L_1-norm). (3) DHP (Li et al., 2020) uses L_1 regularization to induce sparsity, while SRP uses L_2 regularization.

A.2 PERFORMANCE DROP FROM ORIGINAL MODEL

The performance degradation between the original model and the compression model should also be discussed. In Tab. 5, we provide models finetuned for only 1 epoch. Compared to the L_1-norm pruning, the performance on Set5 ($\times$2) drop of our pruned models is significantly smaller.

Pruning ratio	0 (original)	0.1	0.3	0.5	0.7	0.9
L_1-norm	37.99	34.70	33.67	33.83	33.52	27.34
SRP (ours)	37.99	**37.82**	**37.85**	**37.73**	**37.60**	**34.53**

Table 5: PSNR (dB) of EDSR baselines finetuned for only 1 epoch, pruned by L_1-norm vs. our SRP.

A.3 COMPARISONS WITH OTHER PRUNING METHODS FOR IMAGE SR

We compare with other widely used channel pruning methods, like ThiNet (Luo et al., 2017) and DCP (Zhuang et al., 2018). It is hard for us to implement ThiNet in our PyTorch framework from its Caffe code. We implement DCP (Zhuang et al., 2018) for image SR and show results on Set5 ($\times$2) in Tab. 6, where our SRP performs better.

Pruning ratio	0 (original)	0.1	0.3	0.5	0.7	0.9
DCP	37.99	37.92	37.83	37.74	37.60	36.91
SRP (ours)	37.99	**37.97**	**37.89**	**37.84**	**37.71**	**37.28**

Table 6: PSNR of EDSR baseline pruned by DCP and SRP.

A.4 COMPARISONS WITH OTHER SOTA METHODS

We further compare with other recent lightweight and large image SR networks: PAN (Zhao et al., 2020), LatticeNet (Luo et al., 2020), and DRN-L (Guo et al., 2020) (the large model version). In Tab. 7, lightweight models, LatticeNet obtains better than SRP-L with more parameters. Large models, SRPN performs better than DRN-L.

Method	PAN	LatticeNet	DRN-L	SRPN-L	SRPN
Param.	272K	777K	9.8M	623K	15.3M
PSNR (dB)	26.11	26.25	27.03	26.16	**27.25**

Table 7: Model size and PSNR comparisons on Urban100 ($\times$4).

A.5 MODEL COMPLEXITY COMPARISONS FOR LARGE MODELS

We provide model size, FLOPs, and inference time comparison for large models in Tab. 8. We can see our SRPN obtains similar model size, FLOPs with RCAN and SAN, but obtains better performance. When compared with CSNLN, our SRPN performs more efficiently.

A.6 GENERALIZATION OF SRP

We further discuss the generalization ability of our SRP method. (1) SRP can prune *any* kind of block structure. (2) We remove channel attention because we want to remove its effect and observe the effect of SRP better. (3) We apply our SRP to IMDN (Hui et al., 2019) and CARN (Ahn et al., 2018). We provide the results on Set5 ($\times$2) of applying SRP to IMDN and CARN in Tab. 9.

Large models	EDSR	RCAN	SAN	CSNLN	SRPN
Parameters (M)	40.73	15.44	15.67	3.06	15.33
FLOPs (G)	1,042.7	391.9	400.4	2,245.9	391.8
Inference Time (s)	0.37	0.85	1.45	7.14	0.76
PSNR (dB) on Manga109 ($\times$2)	39.10	39.44	39.32	39.37	39.75

Table 8: Param., FLOPs, and time. Input size: $3\times160\times160$.

Pruning ratio	0 (original)	0.1	0.3	0.5	0.7	0.9
CARN+SRP	37.76	37.74	37.65	37.61	37.49	37.05
IMDN+SRP	38.00	37.97	37.89	37.84	37.70	37.36

Table 9: Apply SRP to CARN and IMDN. PSNR on Set5 ($\times$2).

A.7 Discussions with Other Pruning Works

(1) Existing pruning works are mainly for *classification*, while this paper focuses on *image SR*. (2) Pruning methods in classification can hardly apply to SR networks directly, thus we propose a pruning method specifically for SR models. (3) We show comparison with (Hou & Kung, 2020) in Tab. 10. (4) We also discuss the differences with (Kim et al., 2020), which is for *image compression*. We do not think the wisdom of image compression can equally apply to *image SR*. Our results (Tab. 1) suggest the *small-magnitude* filters are *unimportant*, in line with the case in classification.

Small models	Param.	FLOPs	Set5	Set14	B100	Urban100
SRResNet+DI-trim-0.3 (Hou & Kung, 2020)	604K	318.4G	38.05	**33.73**	32.20	32.17
SRPN-L	609K	139.9G	**38.10**	33.70	**32.25**	**32.26**

Table 10: Comparison with (Hou & Kung, 2020) on the same baseline.

A.8 Validation for Random Selection

(1) Random choosing works well in practice because of the proposed regularization scheme. Empirically, it is justified by the strong performance in Tabs. 2 and 3 and Figs. 4 and 5. (2) Results with different random seeds are presented in Tab. 11. We can see that the statistical variation is small, which further shows the robustness of our method.

B More Experiments

B.1 Comparisons with Lightweight Networks

Quantitative Results. We compare our SRPN-L with representative lightweight SR networks: SRCNN (Dong et al., 2014), FSRCNN (Dong et al., 2016), VDSR (Kim et al., 2016a), DRCN (Kim et al., 2016b), CNF (Ren et al., 2017), LapSRN (Lai et al., 2017), DRRN (Tai et al., 2017a), BTSRN (Fan et al., 2017), MemNet (Tai et al., 2017b), SelNet (Choi & Kim, 2017), CARN (Ahn et al., 2018), CARN-M (Ahn et al., 2018), BSRN (Choi et al., 2018), FALSR-A (Chu et al., 2019a), and IMDN (Hui et al., 2019). We show extensive comparisons in Tab. 12.

Visual Results. We also provide visual results in Fig. 6. We can see obviously better results from our SRPN-L than others. The code and visual results of LatticeNet (Luo et al., 2020) are not available. The model size and FLOPs are also much larger than our SRPN-L. We aim to propose a smaller one than IMDN and LatticeNet in terms for both parameters and FLOPs. Consequently, we don't compare with LatticeNet currently. However, our code still supports to train a lightweight network with similar model size and FLOPs as LatticeNet but larger than IMDN.

B.2 Comparisons with Larger Networks

Quantitative Results. Most previous lightweight SR networks neglect to extend their models to deeper and compare with larger networks. As stated in implementation, we extend our SRP method to larger network training and obtain a deeper network SRPN. In Tab. 13, we compare with large SR networks: EDSR (Lim et al., 2017), SRMDNF (Zhang et al., 2018a), DBPN (Haris et al., 2018), RDN (Zhang et al., 2018c), RCAN (Zhang et al., 2018b), NLRN (Liu et al., 2018), RNAN (Zhang

Pruning ratio	0.1	0.3	0.5	0.7	0.9
L_1-norm	37.90 (0.01)	37.80 (0.02)	37.75 (0.03)	37.59 (0.02)	36.88 (0.02)
SRPN (ours)	**37.98 (0.02)**	**37.89** (0.01)	**37.86** (0.02)	**37.70** (0.01)	**37.30** (0.01)

Table 11: Performance and statistical variation with different random seeds.

et al., 2019), SRFBN (Li et al., 2019), OISR (He et al., 2019), SAN (Dai et al., 2019), HAN (Niu et al., 2020), NSR (Fan et al., 2020), IGNN (Zhou et al., 2020), CSNLN (Mei et al., 2020), and RFANet (Liu et al., 2020). We provide performance comparisons in Tab. 13.

Visual Results. We show visual comparisons in Fig. 7. For some very challenging cases, most previous SOTA methods perform similarly and suffer from some blurring artifacts. However, our large model SRPN recovers better results with sharper structures and more faithful details.

B.3 More Plots of Pruning Process.

In Fig. 8, we present more plots of the pruning process of our SRP method on the EDSR_baseline network. There are 16 residual blocks in EDSR_baseline. We evenly select 5 blocks to plot (layer name is texted as title in each sub-plot). The free Conv layers are plotted in the left and constrained Conv layers plotted in the right. There are two observations worth our attention, as follows.

(1) L_1-norm at iteration 0. In free Conv layers, the unimportant (i.e., pruned) filters are selected based on L_1-norm sorting. As a result, at iteration 0, the L_1-norm of pruned filters are always *smaller* than that of kept filters. In comparison, for constrained Conv layers, since the unimportant filters are picked *randomly*, thus the L_1-norm of pruned filters are *close* (occasionally higher due to randomness) to that of kept filters at iteration 0. This observation serves as a "sanity check" to show our algorithm works normally, being consistent with the robustness shown in Tab. 11.

(2) Compensation effect of free/constrained Conv layers. The compensation effect of kept filters (i.e., L_1-norms of these filters go up automatically without any extra regularization – see also Line 756-760 in the main paper) is observed in all plots presented here, to different degrees. For free Conv layers, this trend is clear (the blue solid line always goes up cross different layers); while for constrained Conv layers, the trend is related to the layer location in the network: for low-stage layers (e.g., "model.body.0.body.2", "model.body.2.body.2"), the blue line only goes up a little; while for high-stage layers (e.g., "model.body.12.body.2" and "model.body.15.body.2"), the blue line goes up significantly. We conjecture, this is because the high-stage layers are closer to the output, thus more critical. The low-stage layers do not have to compensate for the pruned filters since the high-stage layers will do this for them. This implies that, probably we should prune fewer filters for the high-stage layers (for now, the pruning ratio is the same across different layers). This issue can potentially bring more performance gains but relatively trivial as compared to the pruning methodology itself, so we do not cover it in the main paper and leave it as a worthy future direction.

B.4 Demo Code

We have attached a demo testing code and pretrained model for our lightweight method SRPN-L with scaling factor $\times 2$. The network structure will be printed in the log file after running the demo code. The demo code is based on EDSR PyTorch code. All the training/testing code, pretrained models, and training/ablation study log files would be available after the paper is published.

Method	Scale	Params	Mult-Adds	Set5		Set14		B100		Urban100	
				PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
SRCNN	×2	57K	52.7G	36.66	0.9542	32.42	0.9063	31.36	0.8879	29.50	0.8946
FSRCNN	×2	12K	6.0G	37.00	0.9558	32.63	0.9088	31.53	0.8920	29.88	0.9020
VDSR	×2	665K	612.6G	37.53	0.9587	33.03	0.9124	31.90	0.8960	30.76	0.9140
DRCN	×2	1,774K	17,974.3G	37.63	0.9588	33.04	0.9118	31.85	0.8942	30.75	0.9133
CNF	×2	337K	311.0G	37.66	0.9590	33.38	0.9136	31.91	0.8962	-	-
LapSRN	×2	813K	29.9G	37.52	0.9590	33.08	0.9130	31.80	0.8950	30.41	0.9100
DRRN	×2	297K	6,796.9G	37.74	0.9591	33.23	0.9136	32.05	0.8973	31.23	0.9188
BTSRN	×2	410K	207.7G	37.75	-	33.20	-	32.05	-	31.63	-
MemNet	×2	677K	2,662.4G	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195
SelNet	×2	974K	225.7G	37.89	0.9598	33.61	0.9160	32.08	0.8984	-	-
CARN	×2	1,592K	222.8G	37.76	0.9590	33.52	0.9166	32.09	0.8978	31.92	0.9256
CARN-M	×2	412K	91.2G	37.53	0.9583	33.26	0.9141	31.92	0.8960	31.23	0.9193
BSRN	×2	594K	1666.7G	37.78	0.9591	33.43	0.9155	32.11	0.8983	31.92	0.9261
FALSR-A	×2	1,021K	234.7G	37.82	0.9595	33.55	0.9168	32.12	0.8987	31.93	0.9256
IMDN	×2	694K	158.8G	38.00	0.9605	33.63	0.9177	32.19	0.8996	32.17	0.9283
SRPN-L	×2	609K	139.9G	38.10	0.9608	33.70	0.9189	32.25	0.9005	32.26	0.9294
SRCNN	×3	57K	52.7G	32.75	0.9090	29.28	0.8209	28.41	0.7863	26.24	0.7989
FSRCNN	×3	12K	5.0G	33.16	0.9140	29.43	0.8242	28.53	0.7910	26.43	0.8080
VDSR	×3	665K	612.6G	33.66	0.9213	29.77	0.8314	28.82	0.7976	27.14	0.8279
DRCN	×3	1,774K	17,974.3G	33.82	0.9226	29.76	0.8311	28.80	0.7963	27.15	0.8276
CNF	×3	337K	311.0G	33.74	0.9226	29.90	0.8322	28.82	0.7980	-	-
DRRN	×3	297K	6,796.9G	34.03	0.9244	29.96	0.8349	28.95	0.8004	27.53	0.8378
BTSRN	×3	410K	176.2G	34.03	-	29.90	-	28.97	-	27.75	-
MemNet	×3	677K	2,662.4G	34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376
SelNet	×3	1,159K	120.0G	34.27	0.9257	30.30	0.8399	28.97	0.8025	-	-
CARN	×3	1,592K	118.8G	34.29	0.9255	30.29	0.8407	29.06	0.8034	28.06	0.8493
CARN-M	×3	412K	46.1G	33.99	0.9236	30.08	0.8367	28.91	0.8000	27.55	0.8385
BSRN	×3	779K	761.1G	34.32	0.9255	30.25	0.8404	29.07	0.8039	28.04	0.8497
IMDN	×3	703K	71.5G	34.36	0.9270	30.32	0.8417	29.09	0.8046	28.17	0.8519
SRPN-L	×3	615K	62.7G	34.47	0.9276	30.38	0.8425	29.16	0.8061	28.22	0.8534
SRCNN	×4	57K	52.7G	30.48	0.8628	27.49	0.7503	26.90	0.7101	24.52	0.7221
FSRCNN	×4	12K	4.6G	30.71	0.8657	27.59	0.7535	26.98	0.7150	24.62	0.7280
VDSR	×4	665K	612.6G	31.35	0.8838	28.01	0.7674	27.29	0.7251	25.18	0.7524
DRCN	×4	1,774K	17,974.3G	31.53	0.8854	28.02	0.7670	27.23	0.7233	25.14	0.7510
CNF	×4	337K	311.0G	31.55	0.8856	28.15	0.7680	27.32	0.7253	-	-
LapSRN	×4	813K	149.4G	31.54	0.8850	28.19	0.7720	27.32	0.7280	25.21	0.7560
DRRN	×4	297K	6,796.9G	31.68	0.8888	28.21	0.7720	27.38	0.7284	25.44	0.7638
BTSRN	×4	410K	165.2G	31.85	-	28.20	-	27.47	-	25.74	-
MemNet	×4	677K	2,662.4G	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630
SelNet	×4	1,417K	83.1G	32.00	0.8931	28.49	0.7783	27.44	0.7325	-	-
CARN	×4	1,592K	90.9G	32.13	0.8937	28.60	0.7806	27.58	0.7349	26.07	0.7837
CARN-M	×4	412K	32.5G	31.92	0.8903	28.42	0.7762	27.44	0.7304	25.62	0.7694
BSRN	×4	742K	451.8G	32.14	0.8937	28.56	0.7803	27.57	0.7353	26.03	0.7835
IMDN	×4	715K	40.9G	32.21	0.8948	28.58	0.7811	27.56	0.7353	26.04	0.7838
SRPN-L	×4	623K	35.8G	32.24	0.8958	28.69	0.7836	27.63	0.7373	26.16	0.7875

Table 12: Quantitative results of lightweight SR networks. Best and second best results are colored with red and blue. Our SRPN-L achieves the best PSNR and SSIM values across all cases.

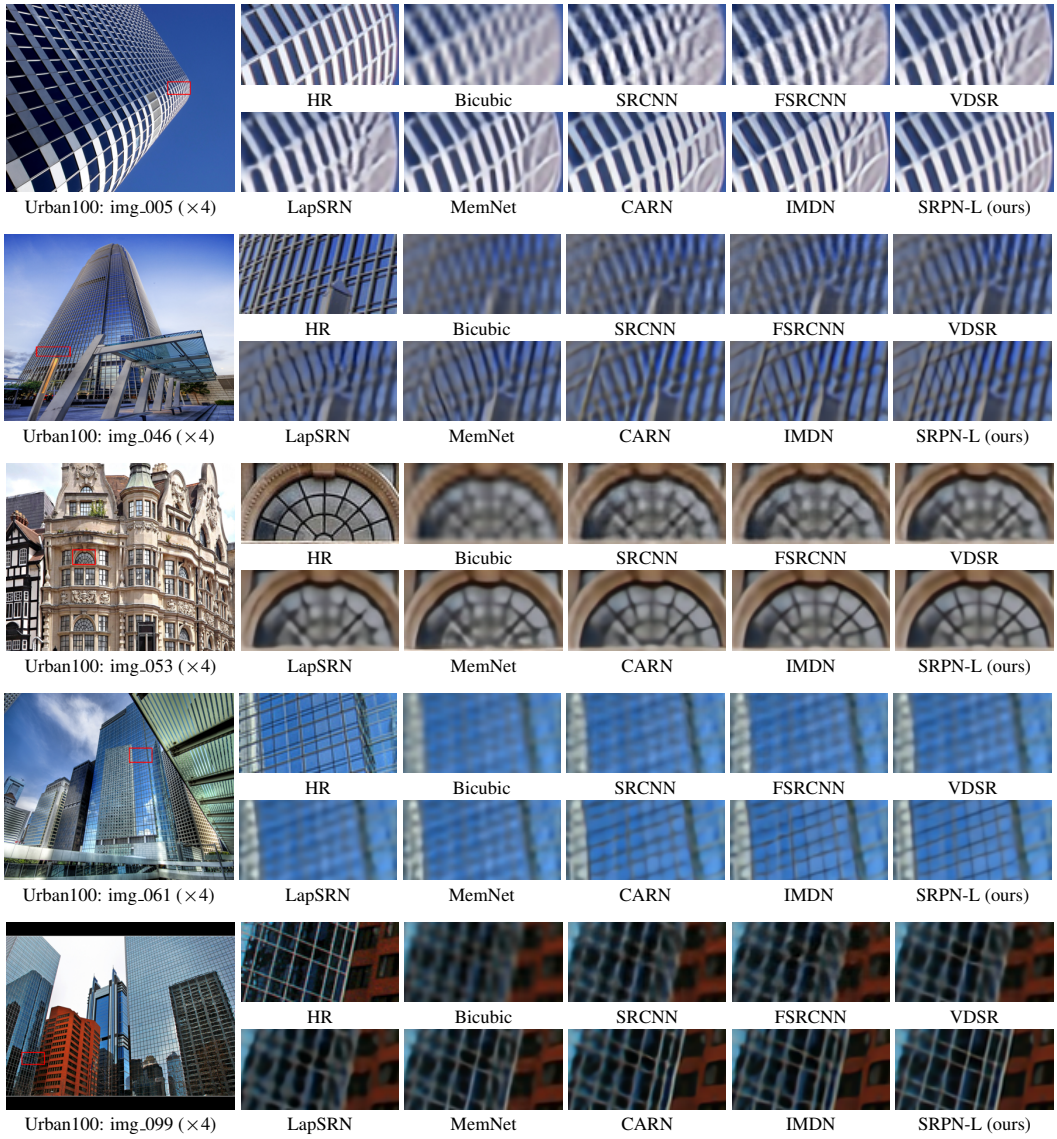

Figure 6: Visual comparison (×4) with lightweight SR networks on Urban100 dataset.

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	×2	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403	30.80	0.9339
EDSR	×2	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
SRMDNF	×2	37.79	0.9601	33.32	0.9159	32.05	0.8985	31.33	0.9204	38.07	0.9761
DBPN	×2	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
RDN	×2	38.24	0.9614	34.01	0.9212	32.34	0.9017	32.89	0.9353	39.18	0.9780
RCAN	×2	38.27	0.9614	34.12	0.9216	32.41	0.9027	33.34	0.9384	39.44	0.9786
NLRN	×2	38.00	0.9603	33.46	0.9159	32.19	0.8992	31.81	0.9249	N/A	N/A
RNAN	×2	38.17	0.9611	33.87	0.9207	32.31	0.9014	32.73	0.9340	39.23	0.9785
SRFBN	×2	38.11	0.9609	33.82	0.9196	32.29	0.9010	32.62	0.9328	39.08	0.9779
OISR	×2	38.21	0.9612	33.94	0.9206	32.36	0.9019	33.03	0.9365	N/A	N/A
SAN	×2	38.31	0.9620	34.07	0.9213	32.42	0.9028	33.10	0.9370	39.32	0.9792
HAN	×2	38.27	0.9614	34.16	0.9217	32.41	0.9027	33.35	0.9385	39.46	0.9785
NSR	×2	38.23	0.9614	33.94	0.9203	32.34	0.9020	33.02	0.9367	39.31	0.9782
IGNN	×2	38.24	0.9613	34.07	0.9217	32.41	0.9025	33.23	0.9383	39.35	0.9786
CSNLN	×2	38.28	0.9616	34.12	0.9223	32.40	0.9024	33.25	0.9386	39.37	0.9785
RFANet	×2	38.26	0.9615	34.16	0.9220	32.41	0.9026	33.33	0.9389	39.44	0.9783
SRPN (ours)	×2	38.34	0.9619	34.29	0.9232	32.47	0.9032	33.50	0.9401	39.76	0.9796
Bicubic	×3	30.39	0.8682	27.55	0.7742	27.21	0.7385	24.46	0.7349	26.95	0.8556
EDSR	×3	34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653	34.17	0.9476
SRMDNF	×3	34.12	0.9254	30.04	0.8382	28.97	0.8025	27.57	0.8398	33.00	0.9403
RDN	×3	34.71	0.9296	30.57	0.8468	29.26	0.8093	28.80	0.8653	34.13	0.9484
RCAN	×3	34.74	0.9299	30.65	0.8482	29.32	0.8111	29.09	0.8702	34.44	0.9499
NLRN	×3	34.27	0.9266	30.16	0.8374	29.06	0.8026	27.93	0.8453	N/A	N/A
RNAN	×3	34.66	0.9290	30.53	0.8463	29.26	0.8090	28.75	0.8646	34.25	0.9483
SRFBN	×3	34.70	0.9292	30.51	0.8461	29.24	0.8084	28.73	0.8641	34.18	0.9481
OISR	×3	34.72	0.9297	30.57	0.8470	29.29	0.8103	28.95	0.8680	N/A	N/A
SAN	×3	34.75	0.9300	30.59	0.8476	29.33	0.8112	28.93	0.8671	34.30	0.9494
HAN	×3	34.75	0.9299	30.67	0.8483	29.32	0.8110	29.10	0.8705	34.48	0.9500
NSR	×3	34.62	0.9289	30.57	0.8475	29.26	0.8100	28.83	0.8663	34.27	0.9484
IGNN	×3	34.72	0.9298	30.66	0.8484	29.31	0.8105	29.03	0.8696	34.39	0.9496
CSNLN	×3	34.74	0.9300	30.66	0.8482	29.33	0.8105	29.13	0.8712	34.45	0.9502
RFANet	×3	34.79	0.9300	30.67	0.8487	29.34	0.8115	29.15	0.8720	34.59	0.9506
SRPN (ours)	×3	34.84	0.9303	30.76	0.8497	29.39	0.8120	29.36	0.8749	34.88	0.9515
Bicubic	×4	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577	24.89	0.7866
EDSR	×4	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.02	0.9148
SRMDNF	×4	31.96	0.8925	28.35	0.7787	27.49	0.7337	25.68	0.7731	30.09	0.9024
DBPN	×4	32.47	0.8980	28.82	0.7860	27.72	0.7400	26.38	0.7946	30.91	0.9137
RDN	×4	32.47	0.8990	28.81	0.7871	27.72	0.7419	26.61	0.8028	31.00	0.9151
RCAN	×4	32.63	0.9002	28.87	0.7889	27.77	0.7436	26.82	0.8087	31.22	0.9173
NLRN	×4	31.92	0.8916	28.36	0.7745	27.48	0.7306	25.79	0.7729	N/A	N/A
RNAN	×4	32.43	0.8977	28.83	0.7871	27.72	0.7410	26.61	0.8023	31.09	0.9149
SRFBN	×4	32.47	0.8983	28.81	0.7868	27.72	0.7409	26.60	0.8015	31.15	0.9160
OISR	×4	32.53	0.8992	28.86	0.7878	27.75	0.7428	26.79	0.8068	N/A	N/A
SAN	×4	32.64	0.9003	28.92	0.7888	27.78	0.7436	26.79	0.8068	31.18	0.9169
HAN	×4	32.64	0.9002	28.90	0.7890	27.80	0.7442	26.85	0.8094	31.42	0.9177
NSR	×4	32.55	0.8987	28.79	0.7876	27.72	0.7414	26.61	0.8025	31.10	0.9145
IGNN	×4	32.57	0.8998	28.85	0.7891	27.77	0.7434	26.84	0.8090	31.28	0.9182
CSNLN	×4	32.68	0.9004	28.95	0.7888	27.80	0.7439	27.22	0.8168	31.43	0.9201
RFANet	×4	32.66	0.9004	28.88	0.7894	27.79	0.7442	26.92	0.8112	31.41	0.9187
SRPN (ours)	×4	32.72	0.9010	29.04	0.7917	27.87	0.7457	27.25	0.8172	31.82	0.9221

Table 13: Quantitative results of large SR networks. Best and second best results are colored with red and blue. Our SRPN obtains the highest PSNR and SSIM values in most cases.

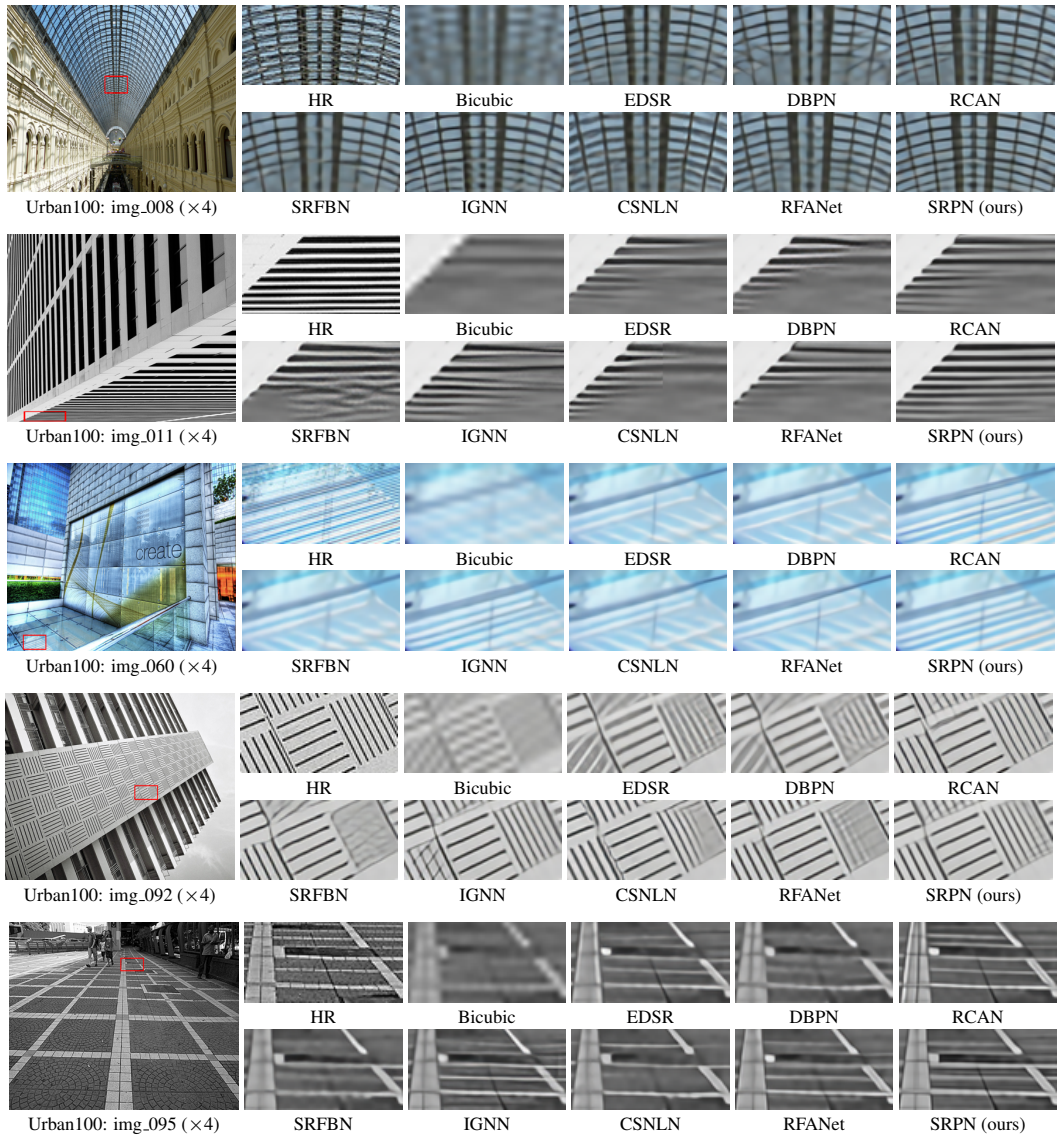

Figure 7: Visual comparison (×4) with large SR networks on Urban100 dataset.

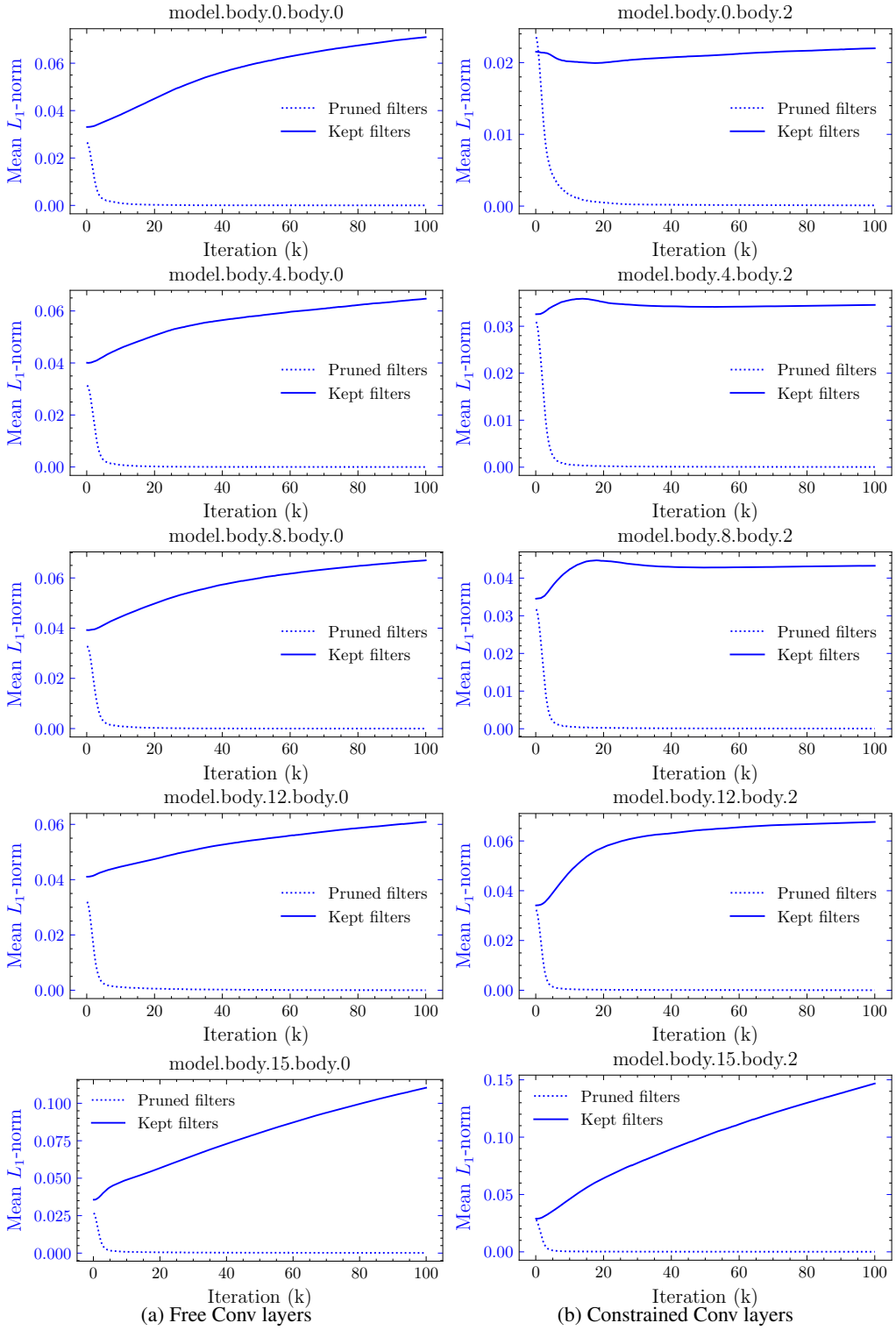

Figure 8: Illustration of the pruning process of five free Conv layers (a) and constrained Conv layers (b) in EDSR_baseline. The mean L_1-norm of pruned or kept filters are plotted across different training iterations. Layer "model.body.12.body.0" and "model.body.12.body.2" have also been presented in the main paper, where we have shown similar observations.