Memory-friendly Scalable Super-resolution via Rewinding Lottery Ticket Hypothesis

Jin Lin[1]*, Xiaotong Luo[1]*, Ming Hong[1], Yanyun Qu[1]†, Yuan Xie[2]†, Zongze Wu[3]
[1]School of Informatics, Xiamen University, Fujian, China
[2]School of Computer Science and Technology, East China Normal University, Shanghai, China
[3]School of Mechatronics and Control Engineering, Shenzhen University, Shenzhen, China
{linjin, xiaotluo, mingh}@stu.xmu.edu.cn, yyqu@xmu.edu.cn,
yxie@cs.ecnu.edu.cn, zzwu@szu.edu.cn

Abstract

Scalable deep Super-Resolution (SR) models are increasingly in demand, whose memory can be customized and tuned to the computational recourse of the platform. The existing dynamic scalable SR methods are not memory-friendly enough because multi-scale models have to be saved with a fixed size for each model. Inspired by the success of Lottery Tickets Hypothesis (LTH) on image classification, we explore the existence of unstructured scalable SR deep models, that is, we find gradual shrinkage sub-networks of extreme sparsity named winning tickets. In this paper, we propose a Memory-friendly Scalable SR framework (MSSR). The advantage is that only a single scalable model covers multiple SR models with different sizes, instead of reloading SR models of different sizes. Concretely, MSSR consists of the forward and backward stages, the former for model compression and the latter for model expansion. In the forward stage, we take advantage of LTH with rewinding weights to progressively shrink the SR model and the pruning-out masks that form nested sets. Moreover, stochastic self-distillation (SSD) is conducted to boost the performance of sub-networks. By stochastically selecting multiple depths, the current model inputs the selected features into the corresponding parts in the larger model and improves the performance of the current model based on the feedback results of the larger model. In the backward stage, the smaller SR model could be expanded by recovering and fine-tuning the pruned parameters according to the pruning-out masks obtained in the forward. Extensive experiments show the effectiveness of MMSR. The smallest-scale sub-network could achieve the sparsity of 94% and outperforms the compared lightweight SR methods.

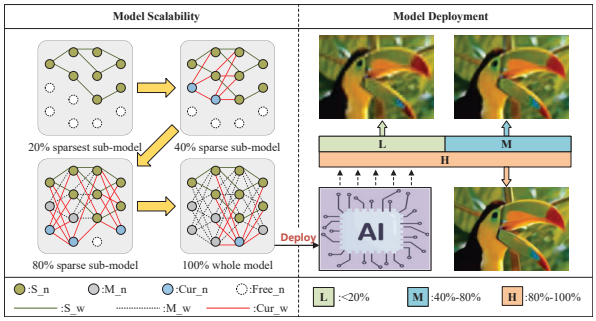

Figure 1. The flowchart of scalable SR network. S_n and S_w denote the neurons and the neural connections (weights) of the simplest subnetwork; M_n and M_w denote the intermediate neurons and neural connections; Cur_n and Cur_w denote the specific neurons and neural connections belonging to the current subnetwork. Free_n denotes the pruning-out neurons. The final model (with 100% recovered parameters) reaches the original size. The scalable model is adjustable to the memory resource allocation.

1. Introduction

Single image super-resolution (SISR) aims to reconstruct a high-resolution (HR) image from the corresponding low-resolution (LR) one. With the rising of deep learning, deep SR methods have made incredible progress. However, the existing SR models mostly require computational and memory resources, so they do not favor resource-limited devices such as mobile phones, robotics, and some edge devices.

The lightweight SR methods are attracting more attention for better application to resource-limited devices. The existing lightweight SR methods mainly focus on designing compact architectures [17, 20] with a fixed size, such as multi-scale pyramid [20], multiple-level receptive fields [17, 18], and recursive learning [19]. However, most lightweight SR models with fixed sizes are not flexible in applications. If one model does not match the resources of

*Equal contribution
†Corresponding authors

the platform, it has to be retrained by compression methods to match the resources and then reloaded onto the devices.

The urgent demand to customize models based on deployment resources is increasing. Dynamic neural networks for SR [14, 22] are proposed to adjust the network architecture according to different computational resources. The existing dynamic deep SR models often explore dynamic depth or width [22, 26], but they either require large memory resources or are not convenient for users to wait for retraining another SR model. The former leads to saving the multi-scale SR models of different sizes and the latter leads to retraining the model before being reused again. The limitation lies in that they are not memory-friendly. In many edge-device SR applications, the devices may be scalable, that is, their memories may be small in the beginning and be expanded later. Thus, we discuss two issues in this paper: 1) how to make a scalable lightweight model for the multi-scale computational recourse. 2) how to make the lightweight model expand to a larger-size model for better performance if the computational recourse is increased.

As for the first issue, inspired by the success of Lottery Ticket Hypothesis [10] which points out that there could exist a properly pruned sub-network named winning tickets to achieve comparable performance against the original dense network in model compression of classification, it is used to find the sub-network for SR. We are the first to study the existence of scalable winning tickets for SR. Iterative pruning and rewinding weights in LTH are beneficial to the scalable lightweight SR model. Iterative pruning may compress the SR model according to an arbitrary size. It is observed in [24] that the winning tickets are related to an insufficient DNN, and rewinding LTH outperforms the original LTH. That is, the initial weights in LTH are replaced with the T-iteration weights during pruning and fine-tuning. The scalable deep SR model is shown in Fig. 1.

As for the second issue, the scalable SR model can customize parameters to adapt to different memory resources rather than load or offload different models for different devices. In other words, during real applications, there will be only one simple model to be employed for inference whose size is decided by the computational resource.

In this paper, we propose a memory-friendly scalable deep SR model (MSSR) via rewinding LTH. We use the rewinding LTH [10] to generate our unstructured scalable mask. MSSR is backtracking and contains forward and backward stages. The former focuses on model compression by rewinding LTH with iterative pruning and fine-tuning, and the latter focuses on iterative model expansion until it goes back to its original size. Multi-scale winning tickets together with the pruning-out masks are obtained by rewinding LTH in the forward stage with the decrease in the number of parameters. The pruning-out masks are nested. In order to make the compressed SR model not degrade significantly, stochastic self-distillation (SSD) is used to improve the representation of the small-scale SR model, and knowledge is transferred from the last-scale model to the current scale model. In the backward stage, the smallest model is expanded gradually to the model with the original size with the expanded mask.

The main contributions of this work are three-fold:

- A memory-friendly scalable dynamic SR lightweight model via rewinding LTH is proposed. MSSR is re-configurable and switchable to sub-networks with different sizes according to on-device resource constraints on the fly.
- MSSR is backtracking, which contains forward and backward stages. Multi-scale winning tickets form nested masks for the multi-scale models. SSD is conducted by replacing the features in randomly selected layers between Teacher and Student to improve the performance of the scalable SR lightweight models.
- Extensive experiments demonstrate that MSSR can generalize to different SR models as well as state-of-the-art attention-based models, ENLCN [1].

2. Related work

2.1. Lightweight Deep Learning-based SR

SISR aims to generate a high-resolution image from the low-resolution version. Recently, efficient and lightweight SISR networks have attracted increasing interest in the computer vision community. The lightweight SR methods consist of static and dynamic neural networks for SR. The earlier lightweight SR methods mainly focus on the static network structure design. IDN [18] utilizes group convolution and combined the local long and short path features effectively. CARN [19] utilizes a recursive cascading mechanism for learning multi-level feature representations. IMDN [17] introduces information multi-distillation blocks for extracting hierarchical features. FALSR [6] applies neural architecture search for efficient network design automatically. LatticeNet [27] designs a lattice block to combine two residual blocks adaptively. However, the aforementioned lightweight SR models need elaborate designs.

Recently, dynamic lightweight SR models are demanded increasingly whose size is adjustable according to different computational resources. DS-Net [22] makes the width of the SR model adjustable, and IKS [26] makes the sparsity of individual kernels adjustable. However, they require saving lightweight SR models with different shrinkage sizes, which is not convenient for users to offload the old model and then load a new one. Our method uses a scalable SR model adjustable to the changing computational recourse to avoid the loading and offloading models.

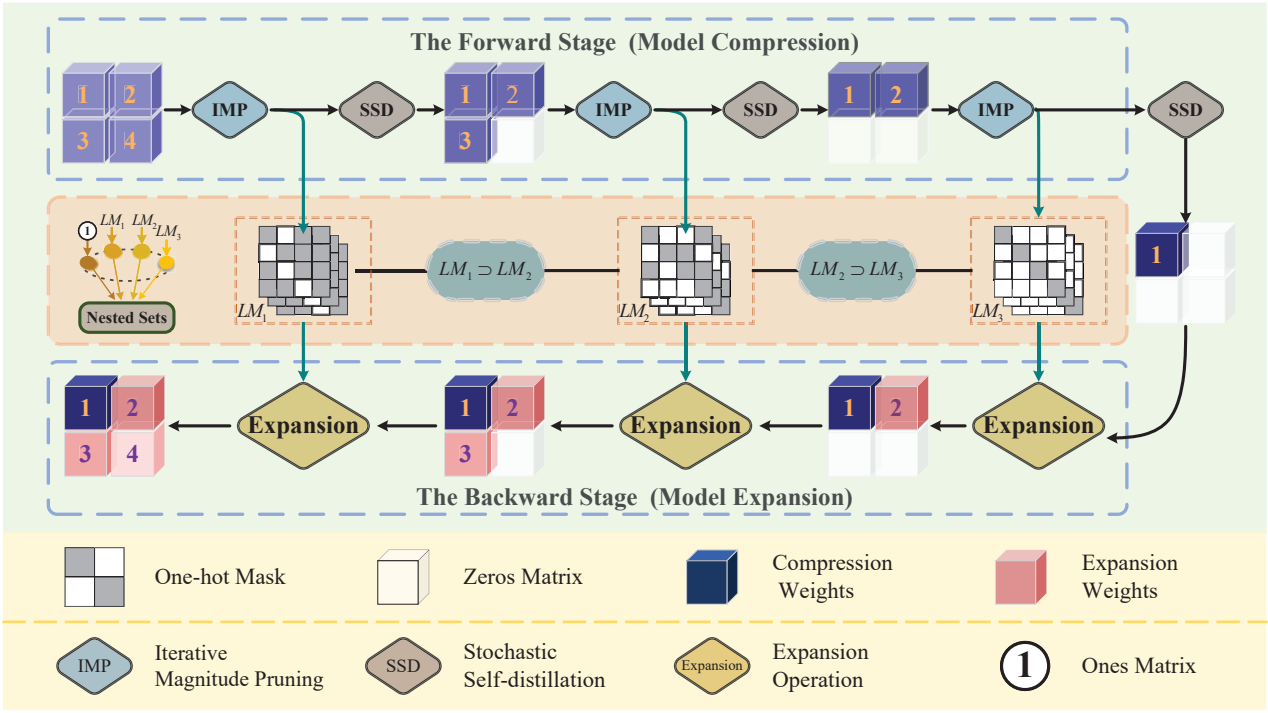

Figure 2. The architecture of memory-friendly scalable deep SR model (MSSR). It contains two stages: the forward stage for model compression and the backward stage for model expansion. In the forward stage, the original SR network is firstly processed by IMP to obtain winning tickets and the pruning-out mask LM_1, and then the winning tickets are processed by SSD and the lightweight subnetwork is obtained. So do the other gradual shrinkage sub-networks and $\{LM_i\}$ forms the nested sets. In the backward stage, according to the pruning-out masks, the smaller lightweight SR model is gradually expanded and finely tuned to obtain a larger lightweight SR model.

2.2. Pruning and Lottery Ticket Hypothesis

Model pruning is to prune away the unimportant connections (parameters) [21] in the neural network which trades off between the model size and performance. The existing pruning methods can be classified into unstructured [6, 13, 21] and structured methods [15, 25, 39]. The structured pruning methods remove the substructures of the model, such as channels, filters, and layers, while the unstructured pruning methods are magnitude-based, where the weights are removed below the threshold. However, magnitude-based pruning methods are not hardware-friendly and computationally efficient. Lottery Ticket Hypothesis (LTH) [10] opened a pandora's box of immense possibilities in the field of pruning and sparse models. It assumes that there exist sparse sub-networks independently training from scratch that can match the performance of dense networks. Some literatures discuss the effectiveness of pruning, fine-tuning [24], and rewinding weights [29] for LTH.

LTH has been widely applied in image classification [9–12, 30, 32, 33], natural language processing [5, 28, 29, 34], and so on. However, few works discuss the existence of LTH in the deep model for image restoration. We are the first to investigate the existence of scalable winning tickets for lightweight deep SR models.

3. Method

3.1. Overview of MSSR

In this section, we introduce the memory-friendly scalable SR model which is an all-in-one model with multi-scale SR models sharing a model. As shown in Fig. 2, our MSSR consists of two stages: the forward stage for model compression, and the backward stage for model expansion.

In the forward stage, we use rewinding LTH (RLTH) to iteratively shrink a dense SR model to obtain a sub-network with the small size not significantly scarifying the performance. During pruning by LTH each time, we treat the current lightweight SR model as Student and its last ancestor lightweight SR model as Teacher, and conduct stochastic self-distillation between Teacher and Student. In Fig. 2, the dark-blue block denotes the winning tickets, and the white block denotes the pruning part. We also record the mask of pruning part and use a matrix with "1" to represent the pruning-out weights and with "0" for others. It is observed that the masks are nested and the initial mask is continuously contained in the later mask. In the backward stage, we iteratively expand the lightweight sub-network according to the saved nested masks. The most lightweight network marked by 1 is expanded iteratively to the gradual less

lightweight network, the expanding order is shown as "1", "1-2 ", "1-2-3", and "1-2-3-4".

3.2. The Forward Stage: Model Compression

The forward streamline is shown in Fig. 3. Let a network denoted by $f(\theta_0; x)$ and the pruned masks which form the "nested sets" denoted by $LM = \{\}$, where θ_0 denotes the initial weights. The flowchart shows a whole process in an iteration for shrinking a model and it contains three components: SR model pre-train, IMP with rewinding weights, and SSD. Firstly, an SR network is trained to reach a steady state, and its weights are named the first non-pruning weights θ. After that, Iterative Magnitude Pruning is conducted on the SR model θ, and the weights below the pre-set threshold are pruned out which forms a pruning mask $m \in \{0, 1\}^n$. Following [24], we rewind the remaining weights to their values in θ, i.e., $\theta' = \theta \odot m$. Then, we retrain the pruned weights θ' by SSD arriving at $\hat{\theta}$. So do the next more lightweight SR model with the model weights $\hat{\theta}$ as the initiation.

LTH with Rewinding for SR. Benefiting from LTHs, we implement iterative magnitude pruning with rewinding to gradually shrink the SR model. The motivation lies in the controllability and flexibility of LTH. We can shrink the model arbitrarily by LTH, which favors the scalable lightweight SR model.

We mainly consider two issues: the model pruning process and the pruning rules. The pruning process corresponds to the process of gradually shrinking the model, and the pruning methods need to meet the following rules: 1) the amount of prune-out parameters is arbitrarily adjustable; 2) the model is shrunk gradually, which leads to the nested sets and the larger model contains its smaller descendant. LTH fits well with the needs of our scalable mask generator and the appropriate use of its task settings to implement the model compression process, in particular discarding its over parameterization, a feature that is not suitable for intensive prediction tasks.

IMP is proven to be effective for LTH. In [24], it is proved that weight rewinding is better than the original weight initiation. That is, unpruned weights are rewound to their values from T-iteration training and retrained from there using the original training schedule.

Compared with one-shot magnitude pruning (OMP) which reaches the goal model at one time, IMP does not shrink a model to reach the pre-set size at one time but shrinks a model with several iterations and the model size is evenly decreased at each iteration. Let the network be pruned for T times and be expected to discard p% of the parameters. The weights/parameters of the network are represented by $\omega \in \mathbb{R}^n$ and the pruning mask by $m \in \{0, 1\}^n$ where n represents all the capacity of the network.

Weight rewinding does not initiate the model with the original initiation weights $\theta_0 \odot m$, but with the t-iteration $\theta_t \odot m$ where $0 < t \leq T$, resulting in weights θ_t and the network function $f(x; \theta_t \odot m)$. We highlight that the original LTH always chooses θ_0 as dense model, while in this work we choose θ_t from the last pruning iteration. After "Pruning" and "Rewinding and Retraining" , $\theta = \hat{\theta}$ is updated and the mask at the t-th iteration is obtained which is denoted as $LM_t = LM_{t-1} \bigcup \{m\}$. In other words, the mask LM_t at t-th iteration is contained in the mask LM_{t-1} at $(t-1)$-th. "Pruning" and "Rewinding and Retraining" are repeated until the model reaches the end condition, which is the iteration steps T or target sparsity p.

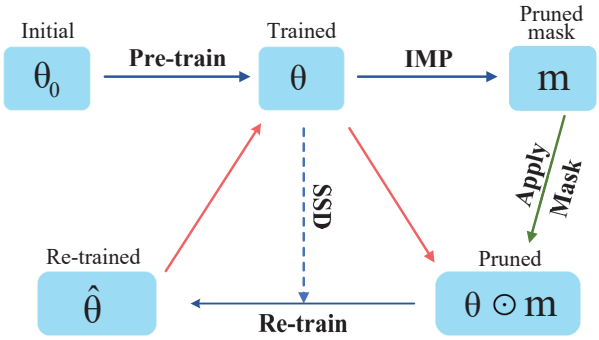

Figure 3. The flowchart of the whole process of IMP and SSD. The initial model θ_0 is pre-trained to obtain a stable model θ. IMP is implemented on θ and the winning ticket (sub-network) is obtained. The sub-network is initiated by $\theta \odot m$ and is processed by SSD for the better model θ.

Stochastic Self-distillation for SR. In the forward stage, due to the progressive shrinking of the SR model, it is definitely difficult to maintain the performance as same as the original model. Especially, when only 1% of the parameters are kept, most magnitude-pruning methods do not work.

In order to mitigate the degradation of lightweight SR models, we adopt self-distillation [36] to boost the performance of the pruned model. Unlike the existing self-distillation which uses the function of features obtained by the model at previous iterations such as the average and summation function, we introduce attribution analysis to self-distillation. We randomly select several layers in Student, and the feature maps of selected layers replace the counterpart of Teacher, which is named Stochastic self-distillation (SSD). Concretely, the current lightweight SR model is regarded as the student network (Student), and its last ancestor network is regarded as the teacher network (Teacher), and Student is obtained by pruning Teacher. SSD is conducted between Teacher and Student. Student is the pruned model of Teacher. Let X and Y denote the network input and ground truth, respectively.

The architecture of SSD is shown in Fig. 4, which contains two streamlines: Teacher and Student. The knowledge in feature maps is transferred from Teacher to Student.

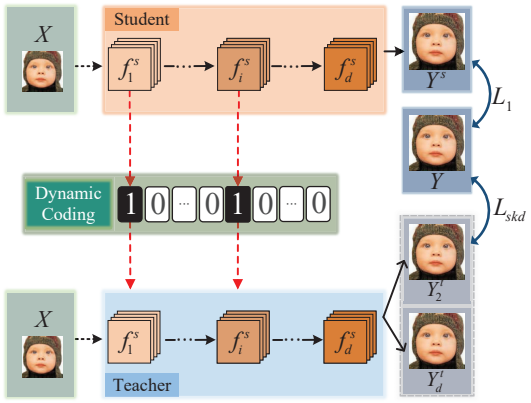

Figure 4. The flowchart of SSD. Student is the sub-network by pruning Teacher. Several layers of Student are selected randomly. The feature maps in a selected layer replace the counterpart of Teacher which outputs the SR result, and so do other selected layers. The total loss is the summation of the knowledge distillation loss L_{skd} and the fidelity loss L_1.

If Student contains d layers, at each-iteration distillation, s layers randomly are selected. In realization, a one-hot vector is randomly generated, named Switch Vector, with a length of d in each training, $\alpha \in \{0,1\}^d$. If the element is equal to 1 in α, the feature maps in the layer corresponding to this position are selected, and are named Lucky feature maps. Then, Lucky feature maps in the selected layers to replace the counterparts of Teacher.

In details, Student firstly is trained for a stable SR model. After that, SSD is conducted. At each iteration, Switch vector is dynamically generated, which is formulated as:

$$\mathrm{DC} = [\alpha_1, \alpha_2, ..., \alpha_d], \tag{1}$$

where α_i denotes the i-th element corresponding to the i-th layer and d is the layer number in Student. Then, lucky feature maps of Student are selected according to α_i, and they are formulated as:

$$\mathrm{LF} = \left[f^s_{\alpha_1}, f^s_{\alpha_2}, ..., f^s_{\alpha_d}\right], \tag{2}$$

where f^s_α denotes the feature obtained from Student. Note that all elements in Switch Vector are independent and identically distributed.

Lucky feature maps replace the counterpart of Teacher each time, and the corresponding output of new Teacher is denoted by $Y^t_{\alpha_i}$, and the corresponding weights of new Teacher are denoted by θ_t, which is formulated as:

$$\begin{cases} Y^t_{\alpha_i} = F(\theta^t; x, f^s_{\alpha_i}), i = 1, ..., d, \\ \mathrm{LY} = \left[Y^t_{\alpha_1}, Y^t_{\alpha_2}, ..., Y^t_{\alpha_d}\right], \end{cases} \tag{3}$$

where LY denotes the set of SR outputs, Y^t_α is the SSD output of lucky maps, f^s_α. All the outputs between Student and Teacher are supervised by HR image Y. The knowledge of Teacher takes more rationality to be distilled into Student.

Therefore, the total loss function utilized to train the SR model consists of the supervised loss L_1 and self-distillation loss L_{skd}, which is formulated as:

$$L_{total} = L_1 + \lambda L_{skd} = \left\|Y^s - Y\right\|_1 + \frac{\lambda}{d}\sum_{i=1}^{d}\alpha_i \left\|Y^t_{\alpha_i} - Y\right\|_1, \tag{4}$$

where λ denotes the regularization parameter, which is empirically set as 0.1 in our experiments. During inference, the SR model θ_p can be customized for various resources according to different sparsity p. After the network training, a lottery mask pool can be built for the backward stage.

Algorithm 1 Memory-friendly scalable lightweight framework for image SR.

Data: HR and LR image datasets: D_{hr}, D_{lr}
Input: Dense network with randomly initial weights: $f(\theta_0)$, pruning rounds: T, and pruning target rate: p
Output: Scalable lightweight SR model

```
/* The forward stage: model compression */
Initializing: Pruned mask m = {1}^n, lottery mask pool LM = {m} and epochs of each training N
Pre-training: Train network for N epochs f(θ0; x) ⟶ f(θ; x)
while i < T − 1 do
    Prune bottom p% × 1/T of m ⊙ θ and update m;
    Re-train the pruned sub-network θ' = m ⊙ θ for N epochs f(θ'; x) ⟶ f(θ̂; x) with SSD(θ, θ'; x);
    Update mask pool LM = LM ⋃ {m} and weights θ = θ̂;
    i ⟵ i + 1;
end while
Generate the sparse sub-network matching the target sparsity, and the mask pool LM;
/* The backward stage: model expansion */
Initializing: Current step t = T and sparsest sub-network θ_T = θ
while t > 1 do
    Calculate the grad mask m = LM_{t−1} − LM_t to determine the trainable weights;
    Fine-tune/re-train the trainable weights for N epochs f(θ_t; x) ⟶ f(θ'_t; x);
    Update θ_t = θ'_t;
    t ⟵ t − 1;
end while
```

3.3. The Backward Stage: Model Expansion

Our intention is to obtain a scalable network that can deploy according to customized memory resources. In this subsection, we introduce how to gradually recover a lightweight SR model, which is a backtracking process of IMP. We start from the most lightweight SR model and end with the original SR model. Corresponding to gradually shrinking the SR model with LTH, we solve two issues in

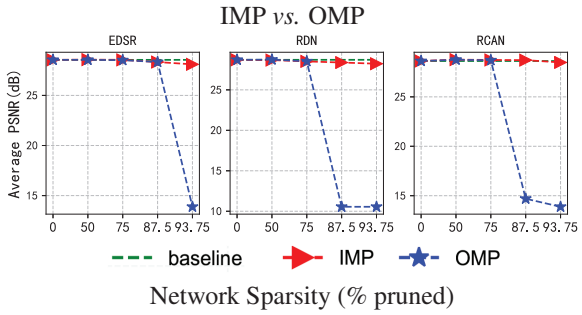

Figure 5. Comparisons of IMP and OMP on various famous SR methods, which are evaluated on Set14 for 4× SR. The performance of the baseline dense network, the sparse sub-network obtained by IMP, as well as the sub-network obtained by OMP. The sub-network is with only 20% of the weights.

the model expansion: iterative model expansion (IME) and the expansion rule, the former refers to the requirement of the IME process and the latter refers to the requirement of the learning rule.

Let's focus on an expansion from the t-th lightweight model to the (t-1)-th lightweight model where t comes from the definition in the forward stage. The larger t is, the smaller model is. During IMP, we obtain two pruned-out masks corresponding to the t-th and (t-1)-th lightweight models, denoted by LM_t and LM_{t-1}, respectively. Their difference represents the expanded positions, i.e., $m = LM_{t-1} - LM_t$. Without changing the t-th model, we recover the weights according to the mask m. After that, we fine-tune the recovered model and obtain the (t-1)-th model. So do the next model expansion. The detailed procedure is described in Algorithm 1.

4. Experiments

4.1. Datasets

DIV2K [2] is adopted to train the SR model, which is a high-quality natural scene dataset widely used for image SR tasks. The whole dataset includes 800 training images and 100 validation images totally with abundant texture and contents. The bicubic degraded LR images are synthesized following the previous works [17, 37]. Besides, the trained models are evaluated on four public SR benchmark datasets: Set5 [4], Set14 [35], B100 [3], and Urban100 [16].

Table 1. Ablation studies about the iterative magnitude pruning (IMP) and the stochastic self-distillation learning (SSD). All these models are validated on Set14 for 4× image SR.

EDSR_baseline	IMP (LTH)	IMP_rewinding	SSD	Sparsity	PSNR
✓	×	×	×	0%	28.55
✓	✓	×	×	94%	27.66
✓	×	✓	×	94%	28.12
✓	×	✓	✓	94%	28.33

Table 2. The performance of the sparse sub-network pruned by IMP-SSD on Set14 for 2×, 3×, 4× image SR.

Scale	Model	Sparsity	PSNR (dB)	Params
2×	EDSR_baseline	**0%**	33.63	1370K
	EDSR_IMP	**94%**	32.88	85K
	EDSR_IMP-SSD	**94%**	33.09 (+0.21)	85K
3×	EDSR_baseline	**0%**	30.30	1558K
	EDSR_IMP	**94%**	29.73	97K
	EDSR_IMP-SSD	**94%**	29.92 (+0.19)	97K
4×	EDSR_baseline	**0%**	28.55	1518K
	EDSR_IMP	**94%**	28.12	95K
	EDSR_IMP-SSD	**94%**	28.33 (+0.21)	95K

4.2. Implementation and Training Details

For training the proposed SR model, 48 × 48 image patches with RGB channels are cropped as the model input. The data augmentation is implemented by random flipping and rotation. The Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ is utilized to train the models. Besides, the mini-batch size is set to 16. The learning rate is initialized as $2e$-4. The pruned weights can be determined by IMP-SSD each 300 epochs and the same setting is adopted in the model expansion stage. We use PyTorch to implement our models with a GTX TITAN GPU.

Objective criteria, i.e., peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) are adopted to evaluate the model performance. The two metrics are both calculated on the Y channel of the YCbCr space.

4.3. Ablation Study

Next, we mainly give a detailed analysis of the proposed lottery-ticket stochastic self-distillation scalable framework (IMP-SSD) for image SR. We adopt EDSR [23] as the backbone network to implement our experiments.

IMP *vs.* OMP. To verify the superiority of IMP over OMP on the compression task of image SR, we conduct IMP and OMP on the forward stage of our method instantiated with three popular SR methods: EDSR [23], RDN [38], and RCAN [37]. As shown in Fig. 5, when the pruned network reaches a very small model size, usually as the sparsity of the model is higher than 80%, where the sparsity means the ratio of the pruning-out parameters, IMP does not significantly scarify the SR performance while OMP degrades a lot. Especially, it obtains 27.66dB in PSNR with 94% sparsity as shown in Table 1. OMP is obviously unable to make the RCAN model reach a very small model similar to that by IMP. That's why we pick IMP as our pruning algorithm, which demonstrates the advantages of LTH once again.

Effectiveness of IMP-SSD. To demonstrate the effectiveness of IMP-SSD, we employ EDSR as the backbone and compare MSSR with EDSR. As shown in Table 2, our IMP-SSD method can not only compress a model to a very

Table 3. Comparison of the quantitative results of average PSNR/SSIM with state-of-the-art efficient SR models.

Scale	Method	Params	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM	DIV2K_valid PSNR/SSIM
4×	EDSR (CVPRW' 2017)	1518K	32.09 / 0.8937	28.55 / 0.7806	27.54 / 0.7348	25.97 / 0.7819	30.38 / 0.8364
	EDSR_IMP-SSD_88% (ours)	190K	31.89 / 0.8911	28.44 / 0.7777	27.46 / 0.7326	25.71 / 0.7743	30.25 / 0.8336
	EDSR_IMP-SSD_94% (ours)	95K	31.74 / 0.8879	28.33 / 0.7749	27.39 / 0.7297	25.48 / 0.7657	30.12 / 0.8303
	RCAN (ECCV' 2018)	15592K	32.19 / 0.8950	28.63 / 0.7819	27.57 / 0.7355	26.21 / 0.7888	30.40 / 0.8331
	RCAN_IMP-SSD_88% (ours)	1949K	32.30 / 0.8963	28.72 / 0.7844	27.63 / 0.7379	26.32 / 0.7932	30.53 / 0.8396
	RCAN_IMP-SSD_94% (ours)	974K	32.02 / 0.8924	28.49 / 0.7788	27.51 / 0.7334	25.87 / 0.7800	30.30 / 0.8347
	RDN (CVPR' 2018)	5579K	32.24 / 0.8965	28.71 / 0.7841	27.64 / 0.7384	26.31 / 0.7928	30.55 / 0.8406
	RDN_IMP-SSD_88% (ours)	697K	31.87 / 0.8902	28.38 / 0.7766	27.42 / 0.7313	25.53 / 0.7681	30.15 / 0.8315
	RDN_IMP-SSD_97% (ours)	174K	31.36 / 0.8826	28.10 / 0.7695	27.23 / 0.7249	25.09 / 0.3116	29.86 / 0.8243
	ENLCN (AAAI' 2022)	1536K	32.14 / 0.8939	28.59 / 0.7812	27.56 / 0.7352	26.05 / 0.7845	30.36 / 0.8360
	ENLCN_IMP-SSD_88% (ours)	192K	31.97 / 0.8911	28.47 / 0.7731	27.47 / 0.7325	25.77 / 0.7313	30.22 / 0.8326
	ENLCN_IMP-SSD_97% (ours)	48K	31.41 / 0.8825	28.09 / 0.7690	27.24 / 0.7250	25.15 / 0.3133	29.84 / 0.8239

Table 4. Comparison between the performance (PSNR) of the sparse sub-networks via IMP-SSD and the performance of the sparse sub-networks of the scalable network (SCL) for 4× SR.

Prune (%)	Set5		Set14		B100		Urban100	
	IMP-SSD	SCL	IMP-SSD	SCL	IMP-SSD	SCL	IMP-SSD	SCL
0 %	32.09	31.94	28.55	28.53	27.54	27.53	25.97	25.89
50 %	32.05	31.96	28.57	28.51	27.55	27.50	25.99	25.79
75 %	32.03	31.92	28.53	28.46	27.52	27.48	25.90	25.73
88 %	31.89	31.82	28.44	28.40	27.46	27.43	25.71	25.61
94 %	31.74	31.74	28.33	28.33	27.39	27.39	25.48	25.48

Table 5. The analysis about IMP-SSD on different methods for 4× image SR validated on Set14.

PSNR / Sparsity / Model	0%	50%	75%	88%	94%	97%	99%
EDSR_IMP-SSD	28.55	28.57	28.53	28.44	28.33	28.05	20.36
RDN_IMP-SSD	28.70	28.68	28.52	28.38	28.23	28.10	27.46
RCAN_IMP-SSD	28.63	28.77	28.73	28.72	28.49	28.15	27.68
ENLCN_IMP-SSD	28.59	28.62	28.56	28.47	28.30	28.09	27.60

small size but also mitigate the degradation in performance for different magnification factors (2×, 3×, and 4×). Besides, the results in Table 1 also illustrate that IMP-SSD further improves the PSNR to 28.33dB with the similar sparsity. Therefore, our method has significant effects on scalable model compression.

IMP-SSD SR *vs*. **Scalable SR.** In order to show the effectiveness of MSSR, we compare the scalable lightweight SR model obtained in the backward stage with the middle result by IMP-SSD in the forward stage. We use EDSR as the backbone, and make the comparison in five degrees of sparsities: 0%, 50%, 75%, 88%, 94%. In Table 4, IMP-SSD generates five pruned SR models corresponding to five sparsities, while scalable SR only has one model. With the increase of sparsity, the SR performance decreases, but the gap between two neighbor sparsities is not more than 0.2dB. Thus, the SR scalable model makes sense. As shown in Table 4, it is observed that scalable SR achieves comparable results to IMP-SSD on four datasets: Set5, Set14, B100, and Urban100. It shows that our method uses only one scalable model that can reach the level of the separately lightweight SR models at different sparse degrees. Therefore, MSSR is memory-friendly.

4.4. Instantiating with Other SR methods

To demonstrate the effectiveness of the proposed scalable lightweight framework (MSSR), we integrate it with four representative SR methods: EDSR [23], RCAN [37], RDN [38], and ENLCN [1]. All these SR models are retrained under the same experimental settings. We show the results in Table 3. It shows that our method is universally applicable to the existing SR methods. Compared with the original models with the magnification factor 4×, under the sparsities of 88% and 94%/97%, the maximum decreases of IMP-SSD instantiated with EDSR, RCAN, RDN, and ENLCN are (0.2dB, −0.1dB, 0.8dB, 0.3dB), and (0.5dB, 0.4dB, 0.9dB, 0.9dB), respectively. Note that, in RCAN, IMP-SSD is better than its original model. We guess that SSD could improve the SR results.

In Table 5, we give more comparisons of the SR performance with more different sparsity. It shows that all methods except EDSR, with the decrease of the model size, our method degrades gradually with not much range of decrease, their ranges are 1.3dB, 1dB, 0.9dB in RCAN, RDN, ENLCN, respectively. EDSR degrades significantly by 8.2dB with the sparsity of 99%.

4.5. Comparison with State-of-the-art

To validate the effectiveness of MSSR, we compare our approach (IMP-SSD) with several state-of-the-art lightweight SR methods: SRCNN [7], FSRCNN [8], CARN [19], and ESRN [31]. Note that IMP-SSD obtains the gradually shrinking lightweight SR models. The comparison results for 2×, 3×, and 4× magnification factors are shown in Table 6. PSNR and SSIM are adopted as metrics to evaluate image quality by convention.

It shows that our IMP-SSD achieves the best results in terms of the coordination of parameters and performance.

Table 6. Comparison of the quantitative results of average PSNR/SSIM with state-of-the-art efficient SR models.

Scale	Method	Params	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM
2×	SRCNN [7] (ECCV' 2014)	57K	36.66 / 0.9542	32.42 / 0.9063	31.36 / 0.8879	29.50 / 0.8946
	FSRCNN [8] (ECCV' 2016)	12K	37.00 / 0.9558	32.63 / 0.9088	31.53 / 0.8920	29.88 / 0.9020
	CARN [19] (ECCV' 2018)	415K	37.68 / 0.9594	33.27 / 0.9149	31.97 / 0.8969	31.18 / 0.9186
	ESRN [31] (AAAI' 2020)	324K	37.85 / 0.9600	33.42 / 0.9161	32.10 / 0.8987	31.79 / 0.9248
	EDSR [23] (CVPRW' 2017)	1370K	37.98 / 0.9605	33.63 / 0.9180	32.17 / 0.8996	32.08 / 0.9280
	EDSR_IMP-SSD_88% (ours)	171K	37.82 / 0.9599	33.44 / 0.9167	32.09 / 0.8987	31.69 / 0.9244
	EDSR_IMP-SSD_94% (ours)	85K	37.51 / 0.9586	33.09 / 0.9137	31.88 / 0.8960	30.94 / 0.9163
3×	SRCNN [7] (ECCV' 2014)	57K	32.75 / 0.9090	29.28 / 0.8209	28.41 / 0.7863	26.24 / 0.7989
	FSRCNN [8] (ECCV' 2016)	12K	33.16 / 0.9140	29.43 / 0.8242	28.53 / 0.7910	26.43 / 0.8080
	CARN [19] (ECCV' 2018)	415K	34.00 / 0.9240	30.07 / 0.8368	28.92 / 0.8007	27.53 / 0.8379
	ESRN [31] (AAAI' 2020)	324K	34.23 / 0.9262	30.27 / 0.8400	29.03 / 0.8039	27.95 / 0.8481
	EDSR [23] (CVPRW' 2017)	1558K	34.37 / 0.9267	30.31 / 0.8409	29.07 / 0.8043	28.05 / 0.8506
	EDSR_IMP-SSD_88% (ours)	195K	34.15 / 0.9252	30.20 / 0.8392	29.00 / 0.8028	27.81 / 0.8449
	EDSR_IMP-SSD_94% (ours)	97K	33.74 / 0.9216	29.92 / 0.8340	28.81 / 0.7976	27.16 / 0.8298
4×	SRCNN [7] (ECCV' 2014)	57K	30.48 / 0.8628	27.49 / 0.7503	26.90 / 0.7101	24.52 / 0.7221
	FSRCNN [8] (ECCV' 2016)	12K	30.71 / 0.8657	27.59 / 0.7535	26.98 / 0.7150	24.62 / 0.7280
	CARN [19] (ECCV' 2018)	415K	31.78 / 0.8894	28.39 / 0.7762	27.42 / 0.7303	25.56 / 0.7674
	ESRN [31] (AAAI' 2020)	324K	31.99 / 0.8919	28.49 / 0.7779	27.50 / 0.7331	25.87 / 0.7782
	EDSR [23] (CVPRW' 2017)	1518K	32.09 / 0.8937	28.55 / 0.7806	27.54 / 0.7348	25.97 / 0.7819
	EDSR_IMP-SSD_88% (ours)	190K	31.89 / 0.8911	28.44 / 0.7777	27.46 / 0.7326	25.71 / 0.7743
	EDSR_IMP-SSD_94% (ours)	95K	31.74 / 0.8879	28.33 / 0.7749	27.39 /0.7297	25.48 / 0.7657

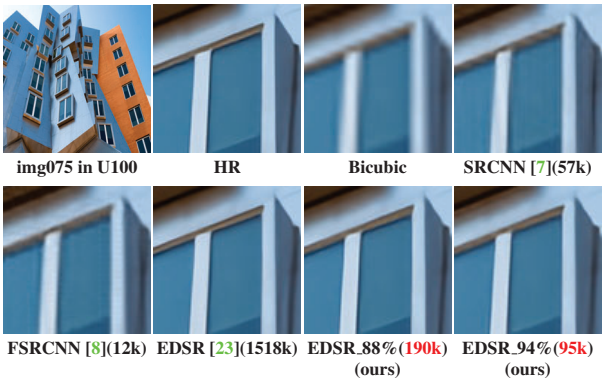

Figure 6. Visual comparisons of IMP-SSD with other lightweight SR models on Urban100 for 4× SR.

In particular, compared with other lightweight SR models such as CARN and ESRN, our IMP-SSD in EDSR with the sparsity of 88% achieves comparable performance while our model size is about half or two-thirds of those. Compared with smaller SR models such as SRCNN and FSRCNN, IMP-SSD with the sparsity of 94%, IMP-SSD achieves significant gains in PSNR by at least 0.4dB, 0.3dB, 0.4dB for 2×, 3×, and 4× SR, respectively. The same conclusion is made in SSIM. Furthermore, compared with the original EDSR, IMP-SSD makes the SR model shrink to reach a very small size but their SR performance does not degrade very much. Compared with the original EDSR, the maximum decreases are 0.4dB, 0.2dB, 0.2dB for IMP-SSD with the sparsity of 88% and 0.6dB, 0.9dB, 0.5dB for IMP-SSD with the sparsity of 94%, respectively. It demonstrates that our method is effective on scalable SR.

We also give the comparison of our method in visual effect in Fig. 6. It is observed that our method achieves a better visual effect than other lightweight SR models even with larger model sparsity.

5. Conclusions

In this paper, we propose a novel memory-friendly scalable lightweight framework (MSSR) with rewinding LTH, which can be deployed on arbitrary computational resources according to customization. MSSR is backtracking and includes two stages: the forward stage for model compression and the backward for model expansion. For the first time, LTH with rewinding weights is implemented to explore the winning tickets of deep SR models. We use IMP to gradually shrink the SR model and not only obtain the scalable lightweight model but also obtain the pruning-out masks which form the nested sets. In order to improve the performance of the shrinking model, SSD is conducted. Unlike the traditional SD, we replace the feature maps on several layers of Teacher with the counterparts of Students and conduct distillation. In the backward stage, we gradually recover the lightweight SR models depending on the pruning-out masks and then fine-tune the current model. We make instantiation on several state-of-the-art SR models. The extensive experimental results are effective.

Acknowledgments. This work is supported by grants from the National Key Research and Development Program of China (No.2020AAA0108301), National Natural Science Foundation of China (No.62176224, No.62222602), and CCF-Lenovo Blue Ocean Research Fund.

References

[1] Efficient non-local contrastive attention for image super-resolution. In *AAAI*, 2022. 2, 7

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017. 6

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 2010. 6

[4] 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

[5] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. In *NeurIPS*, 2020. 3

[6] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. In *ICPR*, 2020. 2, 3

[7] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 2016. 7, 8

[8] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016. 7, 8

[9] Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks. In *ICLR*, 2021. 3

[10] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019. 2, 3

[11] Jonathan Frankle, David J Schwab, and Ari S Morcos. The early phase of neural network training. In *ICLR*, 2020. 3

[12] Sharath Girish, Shishira R Maiya, Kamal Gupta, Hao Chen, Larry S Davis, and Abhinav Shrivastava. The lottery ticket hypothesis for object recognition. In *CVPR*, 2021. 3

[13] 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. 3

[14] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *TPAMI*, 2021. 2

[15] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 3

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

[17] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACM MM*, 2019. 1, 2, 6

[18] Zheng Hui, Xiumei Wang, and Xinbo Gao. Fast and accurate single image super-resolution via information distillation network. In *CVPR*, 2018. 1, 2

[19] Zheng Hui, Xiumei Wang, and Xinbo Gao. Fast and accurate single image super-resolution via information distillation network. In *CVPR*, 2018. 1, 2, 7, 8

[20] 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. 1

[21] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *NeurIPS*, 1989. 3

[22] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *CVPR*, 2021. 2

[23] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017. 6, 7, 8

[24] 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 *ICML*, 2021. 2, 3, 4

[25] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 3

[26] Xiaotong Luo, Mingliang Dai, Yulun Zhang, Yuan Xie, Ding Liu, Yanyun Qu, Yun Fu, and Junping Zhang. Adjustable memory-efficient image super-resolution via individual kernel sparsity. In *ACM MM*, 2022. 2

[27] 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. 2

[28] Sai Prasanna, Anna Rogers, and Anna Rumshisky. When bert plays the lottery, all tickets are winning. In *ACL*, 2020. 3

[29] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *ICLR*, 2020. 3

[30] Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. In *NeurIPS*, 2020. 3

[31] Dehua Song, Chang Xu, Xu Jia, Yiyi Chen, Chunjing Xu, and Yunhe Wang. Efficient residual dense block search for image super-resolution. In *AAAI*, 2020. 7, 8

[32] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020. 3

[33] Shihui Yin, Kyu-Hyoun Kim, Jinwook Oh, Naigang Wang, Mauricio Serrano, Jae-Sun Seo, and Jungwook Choi. The sooner the better: Investigating structure of early winning lottery tickets. In *ICLR*, 2020. 3

[34] Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S Morcos. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. In *ICLR*, 2020. 3

[35] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *ICCV*, 2010. 6

[36] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *ICCV*, 2019. 4

[37] 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*, 2018. 6, 7

[38] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018. 6, 7

[39] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *ECCV*, 2016. 3