Memory-Oriented Structural Pruning for Efficient Image Restoration

Xiangsheng Shi[1,2*], **Xuefei Ning**[1†*], **Lidong Guo**[3*], **Tianchen Zhao**[1], **Enshu Liu**[1], **Yi Cai**[1], **Yuhan Dong**[2], **Huazhong Yang**[1], **Yu Wang**[1†]

[1] Department of Electronic Engineering, Tsinghua University
[2] Shenzhen International Graduate School, Tsinghua University
[3] School of Materials Science and Engineering, Tsinghua University
shi-xs20@mails.tsinghua.edu.cn, foxdoraame@gmail.com, gld21@mails.tsinghua.edu.cn,
suozhang1998@gmail.com, les19@mails.tsinghua.edu.cn, cai-y13@mail.tsinghua.org.cn,
dongyuhan@sz.tsinghua.edu.cn, yanghz@tsinghua.edu.cn, yu-wang@tsinghua.edu.cn

Abstract

Deep learning (DL) based methods have significantly pushed forward the state-of-the-art for image restoration (IR) task. Nevertheless, DL-based IR models are highly computation- and memory-intensive. The surging demands for processing higher-resolution images and multi-task paralleling in practical mobile usage further add to their computation and memory burdens. In this paper, we reveal the overlooked memory redundancy of the IR models and propose a Memory-Oriented Structural Pruning (MOSP) method. To properly compress the long-range skip connections (a major source of the memory burden), we introduce a compactor module onto each skip connection to decouple the pruning of the skip connections and the main branch. MOSP progressively prunes the original model layers and the compactors to cut down the peak memory while maintaining high IR quality. Experiments on real image denoising, image super resolution and low-light image enhancement show that MOSP can yield models with higher memory efficiency while better preserving performance compared with baseline pruning methods.

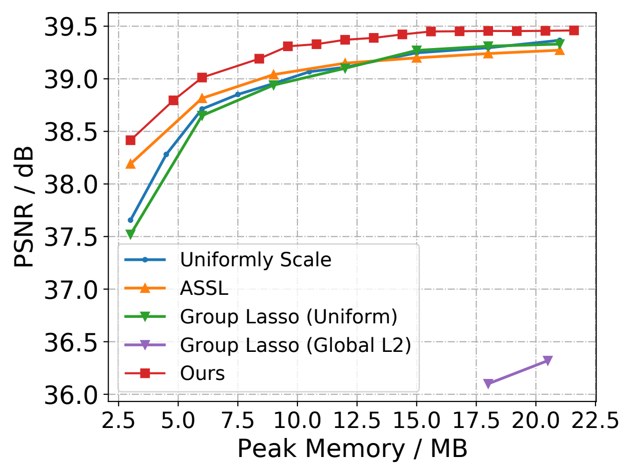

Figure 1: Trade-off between IR performance (PSNR) and peak memory consumption on the SIDD dataset. We compare our pruning method MOSP with uniformly scale, group lasso (Wen et al. 2016) and ASSL (Zhang et al. 2021a).

Introduction

Owing to the adverse environmental conditions and the limitations of image acquisition devices, image degradations (e.g., noise, blur) are often introduced during the image acquisition process. As a fundamental computer vision application, image restoration (IR) aims to recover clean images from such contaminated measurements. Thanks to the superior capacity for learning implicit and generalizable priors in a data-driven fashion, deep learning (DL) based methods (Zamir et al. 2021b; Wang et al. 2021; Chen et al. 2022) have emerged and dominated the field of image restoration, providing satisfactory image reconstructions.

Nevertheless, due to the pixel-to-pixel reconstruction nature of the IR task, these methods have excessive demands for hardware resources in terms of memory and computation. Besides, considering that the computational units on mobile devices need to process multi-tasks simultaneously, the actual resources allocated for IR tasks are further limited. Moreover, the surging demand for processing higher-resolution images (Chen et al. 2018; Lamba and Mitra 2021) adds to the burden of hardware resources. Therefore, deploying the DL-based IR models on resource-constrained mobile devices for practical use is challenging and calls for efficiency improvements.

While improving the computation efficiency of the IR models has been investigated by recent researches, either through compact module design (Chen et al. 2022) or model compression (Zhang et al. 2021a,b), specific optimization for memory consumption of the IR models remains overlooked. In this paper, we point out that **the frequently adopted multi-scale feature aggregation strategy in IR model design is memory-costly and calls for improvements**. Specifically, the widely used U-Net architectures (Chen et al. 2018; Zamir et al. 2021b; Chen et al. 2021) contains multiple long-range skip connections from the encoder to the decoder. Thus the intermediate features need to

*These authors contributed equally.
†Corresponding author.

be stored in memory for a long time before being fused in the decoder. Similarly, short-range skip connections introduced by the residual module design also add to the peak memory consumption. We show that such a feature aggregation strategy is both memory-costly and -redundant. Targeting the contradiction of the tight resource budget on edge devices and the high memory overhead of the multi-scale architectures, our work aims to improve the memory efficiency of IR models.

In this paper, we present Memory-Oriented Structural Pruning (MOSP) to reduce the structural redundancy of IR models. MOSP follows an iterative pruning flow specially designed for peak memory optimization. MOSP first groups layers that co-exist in the memory to form a pruning unit. In each iteration, MOSP selects the layer group that affects the peak memory and cuts down the peak memory by pruning the layers. Besides, the skip connection is a significant resource of the memory burden and cannot be pruned independently. In this regard, MOSP introduces a compactor block to decouple and prune the skip connection properly. As shown in Fig. 1, models pruned by MOSP can achieve better performance-memory efficiency trade-off. The main contributions of this work could be summarized as follows:

- By analyzing the properties of the IR task and multi-scale architectures (e.g., U-Net), we point out that optimizing the peak memory of IR models is important for efficient deployment. Accordingly, we design a Memory-Oriented Structural Pruning (MOSP) framework to improve IR efficiency. MOSP is equipped with an effective grouping strategy that gathers layers with the same temporal memory occupation into a pruning unit. Then, MOSP iteratively prunes the model to meet the memory budget.
- In order to properly compress the skip connections (a major source of the memory burden), we introduce a compactor module onto each skip connection. The compactor decouples the pruning of the skip connection and the main branch, hence the modified skip connections can be pruned independently and incorporated into the corresponding layer groups. Thanks to the enlarged optimization space, better performance-efficiency balance could be achieved.
- Extensive experiments show that the models produced by MOSP could achieve significantly better trade-offs between memory efficiency and performance than models produced by baseline pruning methods. For example, models pruned by MOSP can save up to 50% memory consumption without significant performance degradation and consistently beat those derived by state-of-the-art IR pruning methods.

Related Works

Deep Image Restoration Models

IR task aims to recover clean images from degraded ones (e.g., noise, haze). In recent years, deep learning based methods (Zamir et al. 2020, 2021b; Wang et al. 2021; Zamir et al. 2021a) have achieved great success. They improve the performance in two main directions: **(1)** enhancing the capacity of basic building blocks and **(2)** exploring inter-block connectivity for better information fusion. Chang et al. (Chang et al. 2020) introduce deformable convolution for mining spatial dependence between pixels. RIDNet (Anwar and Barnes 2019) and MIRNet (Zamir et al. 2020) equip basic blocks with attention mechanism. In addition to designing the building block, researchers have also explored the inter-block design space. These methods mainly adopt the U-shaped (Ronneberger, Fischer, and Brox 2015) model and make architectural improvements. MPRNet (Zamir et al. 2021b) and HiNet (Chen et al. 2021) stack U-Nets and integrate information across stages for coarse-to-fine processing. These DL-based restoration methods achieve satisfactory performance. However, their intense computation and memory workload hinder efficient deployment on mobile devices. Therefore, we propose to improve the efficiency of the IR model and focus on reducing its peak memory.

Model Compression

Considerable efforts have been devoted to improving the efficiency of deep neural networks. For instance, network quantization methods (Qiu et al. 2016; Zhou et al. 2016; Esser et al. 2019) seek to yield a highly compact network by reducing its bit width. Network pruning (Han, Mao, and Dally 2015; Wen et al. 2016; Ning et al. 2020a) aims at compressing the model by dropping redundant weights or channels. Various types of Neural Architecture Search (NAS) methods (Zoph and Le 2016; Pham et al. 2018; Ning et al. 2020b) could be applied to automatically design architectures under a certain resource budget (Tan et al. 2019; Cai et al. 2019; Zhang et al. 2022). Although the above-mentioned model compression methods have made significant progress, they are elaborately designed for high-level computer vision tasks. How to adapt them concerning the characteristics of IR tasks remains a question.

Efficient Image Restoration Models

Due to the pixel-to-pixel nature and multi-scale information fusion strategy, IR models suffer from excessive computation and memory workload. In order to alleviate this issue, some researches (Gu et al. 2019; Lamba and Mitra 2021) design efficient architectures. SGN (Gu et al. 2019) adopts a self-guidance strategy to incorporate multi-scale information and extract local features effectively. Another line of research adapts the model compression technique from high-level vision tasks. Li et al. (Li et al. 2020a) quantize super-resolution (SR) models and utilize knowledge distillation to maintain performance. Zhang et al. (Zhang et al. 2021a,b) conduct structured pruning on SR models. The methods mentioned above focus on reducing the FLOPs of the model without considering memory. Differently, we recognize the importance of memory optimization in IR deployment and propose a memory-oriented pruning flow to optimize it accordingly for practical efficiency improvement.

Proposed Method

Fig. 2 shows the two core designs of MOSP: (1) **Skip Connection Compactors.** We point out that much redundancy

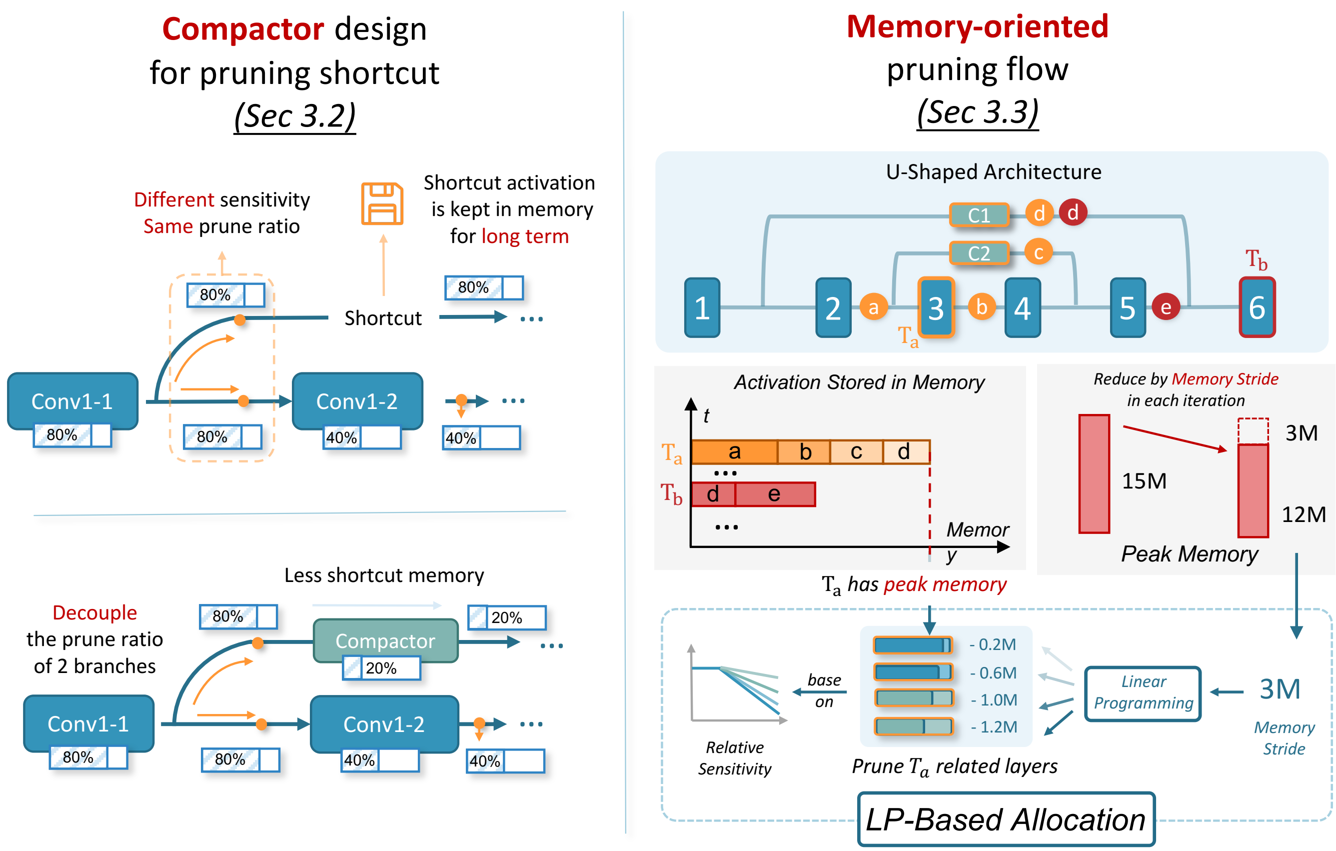

Figure 2: Overall framework of the proposed Memory-Oriented Structured Pruning (MOSP). MOSP is comprised of two major steps. (a) Model adaptation. We substitute the long-term skip connections in U-shaped models with the compactors to decouple them with the corresponding main branch (i.e., consecutive convolution layers). (b) Memory-oriented iterative pruning. We group layers with memory-consumption relevance into a pruning unit, enlarging the optimization space for memory. In each pruning iteration, MOSP first takes an outer step to identify the group with the highest memory usage, and the selected group will get trimmed down by a memory stride. Then MOSP employs an inner step to allocate the memory sparsity within the selected group through linear programming.

resides in the skip-connected features. Therefore, as shown in Fig. 2 (left), we propose to insert *compactors* onto the skip connections to compress the skip-connected features. The compactor design decouples the pruning of the skip-connection branch and the main branch, thus bringing more room for memory optimization. (2) **Memory-Oriented Iterative Pruning Flow.** Fig. 2 (right) shows our memory-oriented iterative pruning flow. This flow explicitly cuts down the peak memory of the given model in each iteration and outputs models satisfying different memory budgets.

Problem Definition for Memory Efficiency

In order to relieve the memory burden for practical deployment, we propose to acquire memory-efficient architectures in the manner of pruning. Specifically, we prune a model to satisfy a provided memory constraint M_c:

$$\min_{\mathbf{W}} \quad \mathcal{L}(\mathcal{F}_{\text{IR}}(X \mid \mathbf{W}; \Theta), Y) \qquad \text{s.t. Memory}(\Theta) \leq M_c \tag{1}$$

where $\mathcal{F}_{\text{IR}}$ denotes the IR model to be pruned, $\mathbf{W}$ denotes the weights of the given model, X, Y denote the input degraded image and the target clean image respectively. One has to identify architecture parameters Θ (i.e., output channels of each layer) to meet the memory budget. After determining the appropriate architecture parameters, the pruned model will be trained to minimize the loss function $\mathcal{L}$.

Compactor Design for Pruning Skip Connections

Multi-scale information fusion has been a fundamental design choice for IR models. To integrate information of multiple scales, the U-shaped architectures need to store intermediate computation results in memory for a long period, bringing heavy memory consumption. Specifically, the original U-Net utilizes four long-term skip connections for information propagation. As can be seen from Fig. 3(a), this design induces large memory consumption.

Then, a natural yet untouched question arises, **do we really need that many features for feature fusion?** Actually,

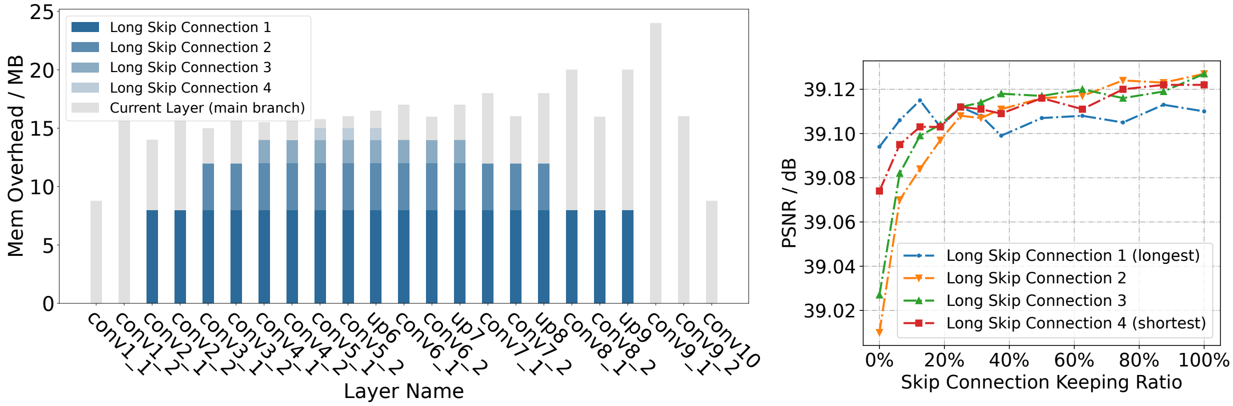

Figure 3: (a) Illustration of memory consumption for each layer in U-Net. Memory consumption is marked with different colors, and estimated by the method described in the "Layer Grouping" section with a single input patch of size $3 \times 256 \times 256$. (b) Performance of U-Nets with respect to different keeping ratios of skip connections. The long skip connection 1 to 4 refer to 4 long-range skip connections of the standard U-Net in descending order according to the length.

our experiments in Fig. 3(b) demonstrate that the answer is **no**. Contrary to the mainstream belief that the skip connections should be faithfully preserved, Fig. 3(b) reveals that the skip connections are highly redundant, implying a broader space for memory optimization.

Intuitively, the features that pass through the main branch might be different from those to be skip-connected: features in the long-range skip connections are rich in low-level information, while those in the main branch are processed to obtain large-scaled information. Hence different keeping ratios of these features should be considered. Nevertheless, the existing pruning methods fail to distinguish such two branches and simply assign the same keeping ratio (usually highly redundant) to the skip connections. For instance, as shown in Fig. 2 (upper left), while the proper keeping ratio of the skip connection may be lower, the skip-connected features are still excessively stored, wasting notable memory.

Therefore, as shown in Fig. 2 (lower left), we propose to **decouple the pruning of the skip connection from that of the main branch by introducing 1×1 convolutions onto the skip connections, i.e., the *compactors***. The benefit of introducing compactors are two-fold. First, features in these two branches can be preserved at different keeping ratios compatible with the contained information. For example, in Fig. 2 (lower left), the keeping ratio of the skip connection branch with the "Compactor" is only 20%, which is more memory-efficient than that forced to share the same keeping ratio with the main branch (Fig. 2 upper left). Second, by introducing only $\sim 1\%$ extra parameters and computations, skip connections can learn to process and preserve features of higher value and thus less affected by pruning.

Memory-Oriented Pruning Flow

To get a solution for the problem defined in Eqn. 1, we propose a memory-oriented pruning flow (see Appendix for detailed algorithm). Specifically, before the pruning process, we first analyze the memory dependency pattern of the original architecture and build the "pruning groups" of layers. Each pruning group contains the layers whose features need to co-exist in the memory simultaneously. The overall pruning process contains outer loops and inner loops. In each outer iteration, we compress the pruning group with the current highest memory cost, cutting down its memory cost by m_o (the outer memory stride). To decide the concrete keeping ratio for each layer in this group, we conduct an inner loop of linear programming optimizations. Each linear programming problem in the inner loop aims to minimize the performance degradation while satisfying the requirement of reducing the memory by m_s (the inner memory stride).

Layer Grouping. We devise a group-wise pruning granularity in terms of memory occupation relevance. Specifically, we group layers co-existing in the memory into a pruning unit. For instance, in Fig. 2, during the computation of M_3 (time step T_a), the input features, the output features, and two parallel skip-connected features (point a, b, c and d in Fig. 2, respectively) are stored in the memory simultaneously. Accordingly, we can prune M_2, M_3, C_1, C_2 by removing some of the output filters to reduce the memory overhead at this time step. Or in another word, these four layers are in the pruning group corresponding to time step T_a. The memory consumption of the pruning group is then estimated as the sum of the memory taken by the output features of the corresponding layers, while the memory taken by layer parameters is neglected. Note that pruning groups corresponding to different time steps can have intersections, i.e., one layer could be in multiple pruning groups. For example, C_1 is in both the T_a and T_b pruning groups in Fig. 2.

Outer Loop. In each outer iteration, we first evaluate the current memory consumption of each group. The group with the highest memory overhead is selected for pruning. As many intersections exist between pruning groups, layer pruning schemes obtained by directly pruning the selected group to meet the final memory constraint M_c may be suboptimal for other groups. Instead, we take iterative steps (i.e., the outer loops) to satisfy the final memory constraint gradually. In each outer loop, we cut down the selected group's memory by m_o (the outer memory stride, 3MB in Fig. 2). In this way, we can better balance memory reduction between different groups and thus get elaborately determined pruning schemes. The outer loop is shown in Algo. line 2 to 17. After determining the group to prune, we carry out an inner loop to get a finer pruning scheme inside the selected group. At the end of each outer iteration, we finetune the model for a short period (e.g., one epoch) to help the model adapt to the channel reduction. The outer loop ends when all groups respect the memory constraint M_c.

Inner Loop. In each inner loop (Algo. line 5 to 11), we are supposed to determine the amount of the filters to prune for each layer in the group to satisfy the memory pruning stride m_o while maintaining the model performance. We modify the optimization problem defined in Eqn.1 as follows:

$$\begin{aligned} \min_{\boldsymbol{\theta}_g^t} \quad & \mathcal{L}(\mathcal{F}_{\text{IR}}(X \mid \mathbf{W}; \boldsymbol{\theta_g^t}), Y) \\ & - \mathcal{L}(\mathcal{F}_{\text{IR}}(X \mid \mathbf{W}; \boldsymbol{\theta_g^{t-1}}), Y) \qquad \text{(2a)} \\ \approx \min_{\boldsymbol{\theta}_g^t} \quad & \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta_g}} \left(\boldsymbol{\theta}_g^t - \boldsymbol{\theta}_g^{t-1}\right) \qquad \text{(2b)} \\ \approx \min_{\boldsymbol{\theta}_g^t} \quad & \boldsymbol{s}_g^t \triangle \boldsymbol{\theta}_g^t \qquad \text{(2c)} \\ \text{s.t.} \quad & \boldsymbol{m}_g \, \boldsymbol{\theta}_g^t \geq m_o. \end{aligned}$$

As shown in Eqn. 2a, at outer loop t, we need to identify the pruning ratios of layers in the selected group g, i.e., $\boldsymbol{\theta}_g^t$, that minimizes the performance degradation compared with the model pruned in the last loop. We use a linear approximation of the objective and thereby simplify the original problem into a linear programming (LP) problem with a constraint ($\boldsymbol{m}_g$ in the constraint is a vector denoting the per-channel memory overhead of the layers in group g).

We use a simple numeric differentiation method to fit $\boldsymbol{s}_g^t = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta_g}}|_{\boldsymbol{\theta}_g^{t-1}}$. Specifically, for each layer in the current group, we prune the layer by one to a few filters and evaluate the consequent performance degradation (i.e., sensitivity analysis). Then we linear fit the performance degradation with the pruned filters of the layer. The resulting sensitivity vector, $\boldsymbol{s}_g^t$, represents the degree to which the overall performance is affected by the layer. After obtaining the linear sensitivity coefficients, we conduct LP to decide the concrete pruning scheme of the grouped layers.

Since we adopt a local linear approximation fashion, it is critical to guarantee sufficient local linearity for accurate fitting. In practice, we further split the outer memory stride m_o into several smaller strides m_s in inner loops and take the above steps to decide a finer pruning scheme $\boldsymbol{\theta}_g^k$, that can be accumulated into the final scheme for this outer loop:

$$\boldsymbol{\theta}_g^t = \sum_{k=1}^{K} \boldsymbol{\theta}_g^k, \qquad (3)$$

where k denotes the inner step k. This split helps guarantee better local linearity, and thus more accurate fitting.

Experiments

Experimental Settings

We conduct experiments on image denoising, super resolution (SR) and low-light enhancement tasks to validate the effectiveness of the proposed method. We provide the implementation details and results of image denoising experiments in this section. Please refer to Appendix for settings and results of SR and low-light enhancement tasks.

Datasets and Evaluation. For real image denoising, we use 320 high-resolution images in the SIDD dataset (Abdelhamed, Lin, and Brown 2018) as the training data. And we report the results evaluated on 1,280 validation patches in SIDD. We perform quantitative comparisons using the PSNR and SSIM (Wang et al. 2004) metrics. When estimating the peak memory of a model, we fix the size of the image as 2K (i.e., 2048 $\times$ 1080).

Backbone and Baseline methods. We compare MOSP with various pruning methods, including uniformly scale, global L2 pruning, group lasso (Wen et al. 2016) , S-Net (Yu and Huang 2019b), US-Net (Yu and Huang 2019c), AutoSlim (Yu and Huang 2019a), Random Search (Li et al. 2022) and DHP (Li et al. 2020b). We also compare with ASSL (Zhang et al. 2021a) that has considered the IR task properties. We choose the standard U-Net (Chen et al. 2018) and its variant Res-U-Net (Wang et al. 2021) as pruning baselines for their representative multi-scale structures. The latter contains both long and local skip connections.

Implementation Details. The models are trained on 256×256 patches with a batch size of 32. Random horizontal and vertical flips are applied to the training patches as data augmentation. We use Adam optimizer (Kingma and Ba 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$. In the pretrain stage, we train the baseline models for 80 epochs, with the initial learning rate set as 1×10^{-4} and decreased to half every 20 epochs. For MOSP, we additionally insert compactors onto the skip connections in the pretrained models and finetune for extra 10 epochs. We finetune the pruned models for 20 epochs to help the models adapt to the channel reduction. During the finetune stage, the learning rate is set to 1×10^{-4} and decreased to 5×10^{-5} after 10 epochs. As for MOSP hyper-parameters, the outer memory stride is by default 2MB and the inner memory step is set to be the highest per-channel memory in the current selected group.

Comparisons on Real Image Denoising

We report IR performance as well as the resource overheads of the models obtained by different pruning methods in Tab.1. Compared with the methods that individually prune each layer and neglect the significance of skip connection pruning, the models pruned by our method consistently achieve the best performance in PSNR and SSIM under different memory constraints. Especially, while saving $4\times$ memory, the model pruned by MOSP only compromises less than 1% PSNR. As shown in Fig. 1, MOSP achieves the best IR performance - memory efficiency trade-off among

Method	Peak Memory	Params	Performance PSNR	SSIM
0.25 × scale	192M	0.48M	38.71	0.911
Global L2	192M	2.40K	29.00	0.851
Group Lasso	192M	0.48M	29.16	0.853
S-U-Net	192M	7.75M	37.92	0.909
US-U-Net	192M	7.75M	37.94	0.909
AutoSlim	192M	0.99K	29.10	0.840
DHP	188M	1.04M	38.58	0.915
Random Search	184M	0.38M	38.39	0.911
ASSL	192M	0.48M	38.81	0.916
MOSP (Ours)	192M	7.27M	**38.92**	**0.916**
0.5 × scale	384M	1.94M	39.11	0.917
Global L2	384M	0.02M	35.03	0.868
Group Lasso	384M	1.94M	34.81	0.86
S-U-Net	384M	7.75M	38.81	0.916
US-U-Net	384M	7.75M	38.24	0.910
AutoSlim	384M	7.80K	34.81	0.862
DHP	368M	2.18M	38.89	0.917
Random Search	352M	1.76M	38.98	0.867
ASSL	384M	1.94M	39.14	0.919
MOSP (Ours)	384M	7.76M	**39.37**	**0.922**
0.75 × scale	576M	4.36M	39.29	0.919
Global L2	576M	0.07M	36.08	0.886
Group Lasso	576M	4.36M	36.10	0.887
S-U-Net	576M	7.75M	39.15	0.918
US-U-Net	576M	7.75M	38.85	0.915
AutoSlim	576M	0.02M	35.22	0.867
DHP	600M	4.64M	39.17	0.920
Random Search	576M	4.20M	39.12	0.917
ASSL	576M	4.36M	39.24	0.919
MOSP (Ours)	576M	7.83M	**39.45**	**0.922**
Unpruned	768M	7.75M	39.47	0.921

Table 1: U-Net pruning results on the SIDD dataset. The pruning results of Res-U-Nets are provided in the Appendix.

the compared methods. The above results indicate that benefit from the enlarged memory optimization room and dedicated iterative memory pruning flow, MOSP has a better capacity of balancing memory budget across the layers. The pruning results of Res-U-Nets are provided in Appendix.

The visual results of pruned U-Nets with 50% original memory overheads are illustrated in Fig. 4. It can be seen that models produced by MOSP is capable of better restoring structural content, while those by compared methods fail to completely remove noise, or overly smooth the contents. The overall results demonstrate that under the same memory budgets, models pruned by MOSP can achieve both quantitatively and qualitatively better results.

Pruning Pattern

We then analyze the pruning patterns produced by the compared methods and MOSP. Visualization results can be found at Appendix. The findings are two-fold:

- Skip connections have larger pruning ratios than main branches, which demonstrates the high memory redundancy induced by skip connections and thus the importance of introducing compactors. By contrast, the skip connections and the main branch of models pruned by the compared methods share the same keeping ratios, leaving the memory redundancy unexplored.
- ASSL prunes layers to the same ratio, hence fail to distinguish layers with different memory efficiency-performance trade-offs, leading to suboptimal results. Global L2 and AutoSlim tend to prune out the layers in the middle of the model before turning to slim down the layers aside, which destroys the structural integrity of the IR models. MOSP considers both the memory resource and layers' sensitivity and therefore can obtain higher memory efficiency and more balanced pruning patterns.

Ablation Study

In this section, we present the ablation studies of the proposed MOSP framework and analyze the effect of each design choice. We adopt U-Net as baseline model in each ablation experiment. We also provide the practical peak memory

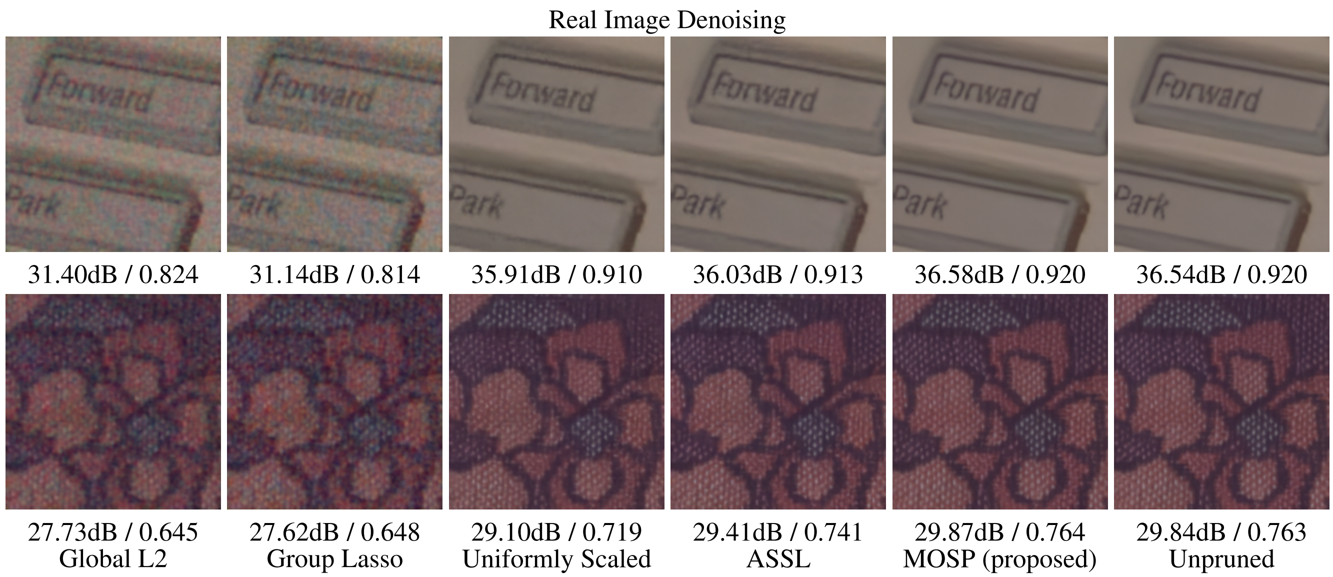

Figure 4: Qualitative comparisons between the existing pruning methods and the proposed method. Memory consumption of the models are pruned to the half. Patches are from SIDD validation set. PSNR and SSIM scores are attached below.

m_o/MB	1	2	4	8	12
PSNR	39.371	39.370	39.360	39.339	39.283

Table 2: Comparison on the memory strides of outer loops.

m_s/MB	0.5	1	1.5	2.0
PSNR	39.386	39.384	39.383	39.383

Table 3: Comparison on the memory steps of inner loops.

measurements across different platforms in Appendix.

The effect of the compactor. Fig. 5 shows the performance of the pruned models with and without compactors. With the help of skip-connection compactors, the memory redundancy can be explored and the performance is significantly improved under different peak memory constraint, ranging from 21.6MB to 4.8MB (90% to 20% of original memory overheads). Especially, when the peak memory of the model is compressed to 20%, the pruned model with compactors outperforms that without compactors by nearly 1dB, which proves the effectiveness of the compactors.

The effect of outer stride m_o and inner steps m_s. Directly pruning the selected layer group to satisfy the memory constraint in each outer loop can notably shorten the whole pruning progress. However, pruning with such a huge memory stride overlooks to balance memory reduction between the pruning groups and only leads to suboptimal results, as there exist many intersections between different pruning groups. Tab. 2 shows the pruning results of different memory stride m_o. The smaller the memory stride, the higher the accuracy. Nevertheless, smaller memory strides result in longer process time because it takes more outer loops to finish the whole pruning procedure. We further study the effect of the inner memory step. As shown in Tab. 3, though the result difference is small in this experiment, we preserve the inner loop, which aims at keeping the local linearity, to provide a more general method formulation.

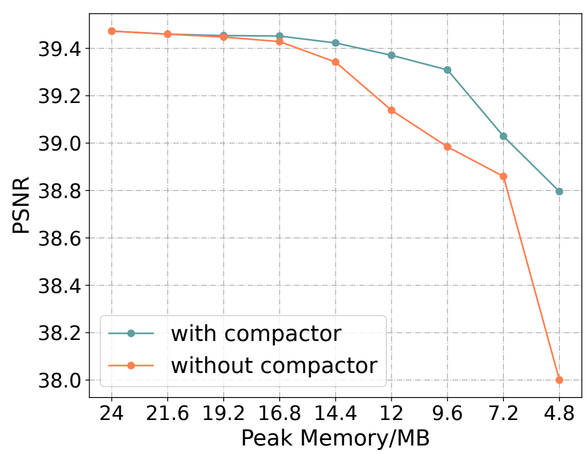

Figure 5: Comparison results between the settings with and without compactor. Tested on the SIDD validation patches.

Conclusion

In this paper, we propose an iterative pruning flow, MOSP, specially designed for peak memory optimization. MOSP introduces skip connection compactors and reduces the memory redundancy induced by the multi-scale structure appropriately. Compared with the previous pruning methods, the proposed method derives models with better image restoration performance under similar memory budget.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. U19B2019, 62104128, U21B2031, 61832007), National Key Research and Development Program of China (No. 2019YFF0301500), Tsinghua EE Xilinx AI Research Fund, and Beijing National Research Center for Information Science and Technology (BNRist). We thank Prof. Yunxin Liu and Dr. Yuzhi Wang for their valuable discussions.

References

Abdelhamed, A.; Lin, S.; and Brown, M. S. 2018. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1692–1700.

Anwar, S.; and Barnes, N. 2019. Real image denoising with feature attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 3155–3164.

Bevilacqua, M.; Roumy, A.; Guillemot, C.; and Alberi-Morel, M. L. 2012. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*.

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2019. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*.

Chang, M.; Li, Q.; Feng, H.; and Xu, Z. 2020. Spatial-adaptive network for single image denoising. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 171–187. Springer.

Chen, C.; Chen, Q.; Xu, J.; and Koltun, V. 2018. Learning to see in the dark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3291–3300.

Chen, L.; Chu, X.; Zhang, X.; and Sun, J. 2022. Simple Baselines for Image Restoration. *arXiv preprint arXiv:2204.04676*.

Chen, L.; Lu, X.; Zhang, J.; Chu, X.; and Chen, C. 2021. HINet: Half instance normalization network for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 182–192.

Esser, S. K.; McKinstry, J. L.; Bablani, D.; Appuswamy, R.; and Modha, D. S. 2019. Learned step size quantization. *arXiv preprint arXiv:1902.08153*.

Gu, S.; Li, Y.; Gool, L. V.; and Timofte, R. 2019. Self-guided network for fast image denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2511–2520.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Huang, J.-B.; Singh, A.; and Ahuja, N. 2015. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5197–5206.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lamba, M.; and Mitra, K. 2021. Restoring Extremely Dark Images in Real Time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3487–3497.

Li, H.; Yan, C.; Lin, S.; Zheng, X.; Zhang, B.; Yang, F.; and Ji, R. 2020a. Pams: Quantized super-resolution via parameterized max scale. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 564–580. Springer.

Li, Y.; Adamczewski, K.; Li, W.; Gu, S.; Timofte, R.; and Van Gool, L. 2022. Revisiting Random Channel Pruning for Neural Network Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 191–201.

Li, Y.; Gu, S.; Zhang, K.; Gool, L. V.; and Timofte, R. 2020b. Dhp: Differentiable meta pruning via hypernetworks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 608–624. Springer.

Lim, B.; Son, S.; Kim, H.; Nah, S.; and Mu Lee, K. 2017. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition workshops (CVPRW)*, 136–144.

Martin, D.; Fowlkes, C.; Tal, D.; and Malik, J. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, volume 2, 416–423. IEEE.

Ning, X.; Zhao, T.; Li, W.; Lei, P.; Wang, Y.; and Yang, H. 2020a. DSA: More Efficient Budgeted Pruning via Differentiable Sparsity Allocation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Ning, X.; Zheng, Y.; Zhao, T.; Wang, Y.; and Yang, H. 2020b. A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Qiu, J.; Wang, J.; Yao, S.; Guo, K.; Li, B.; Zhou, E.; Yu, J.; Tang, T.; Xu, N.; Song, S.; Wang, Y.; and Yang, H. 2016. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In *FPGA '16*.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention (MICCAI)*, 234–241. Springer.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. V. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2820–2828.

Timofte, R.; Agustsson, E.; Van Gool, L.; Yang, M.-H.; and Zhang, L. 2017. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition workshops (CVPRW)*, 114–125.

Wang, Z.; Bovik, A.; Sheikh, H.; and Simoncelli, E. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612.

Wang, Z.; Cun, X.; Bao, J.; and Liu, J. 2021. Uformer: A general u-shaped transformer for image restoration. *arXiv preprint arXiv:2106.03106*.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 29.

Yu, J.; and Huang, T. 2019a. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*.

Yu, J.; and Huang, T. S. 2019b. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1803–1811.

Yu, J.; and Huang, T. S. 2019c. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1803–1811.

Zamir, S. W.; Arora, A.; Khan, S.; Hayat, M.; Khan, F. S.; and Yang, M.-H. 2021a. Restormer: Efficient Transformer for High-Resolution Image Restoration. *arXiv preprint arXiv:2111.09881*.

Zamir, S. W.; Arora, A.; Khan, S.; Hayat, M.; Khan, F. S.; Yang, M.-H.; and Shao, L. 2020. Learning enriched features for real image restoration and enhancement. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 492–511. Springer.

Zamir, S. W.; Arora, A.; Khan, S.; Hayat, M.; Khan, F. S.; Yang, M.-H.; and Shao, L. 2021b. Multi-stage progressive image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14821–14831.

Zeyde, R.; Elad, M.; and Protter, M. 2010. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, 711–730. Springer.

Zhang, L. L.; Han, S.; Wei, J.; Zheng, N.; Cao, T.; and Liu, Y. 2022. Nn-METER: Towards Accurate Latency Prediction of DNN Inference on Diverse Edge Devices. *GetMobile: Mobile Comp. and Comm.*, 25(4): 19–23.

Zhang, Y.; Wang, H.; Qin, C.; and Fu, Y. 2021a. Aligned Structured Sparsity Learning for Efficient Image Super-Resolution. *Advances in Neural Information Processing Systems (NIPS)*, 34.

Zhang, Y.; Wang, H.; Qin, C.; and Fu, Y. 2021b. Learning Efficient Image Super-Resolution Networks via Structure-Regularized Pruning. In *International Conference on Learning Representations (ICLR)*.

Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; and Zou, Y. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.

Zoph, B.; and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.