Attentive Fine-Grained Structured Sparsity for Image Restoration

Junghun Oh[1] Heewon Kim[1] Seungjun Nah[1,3] Cheeun Hong[1] Jonghyun Choi[4] Kyoung Mu Lee[1,2]

[1]Dept. of ECE, ASRI, [2]IPAI, Seoul National University [3]NVIDIA [4]Yonsei University

[1]{dh6dh, ghimhw, cheeun914, kyoungmu}@snu.ac.kr, [3]seungjun.nah@gmail.com, [4]jc@yonsei.ac.kr

Abstract

Image restoration tasks have witnessed great performance improvement in recent years by developing large deep models. Despite the outstanding performance, the heavy computation demanded by the deep models has restricted the application of image restoration. To lift the restriction, it is required to reduce the size of the networks while maintaining accuracy. Recently, N:M structured pruning has appeared as one of the effective and practical pruning approaches for making the model efficient with the accuracy constraint. However, it fails to account for different computational complexities and performance requirements for different layers of an image restoration network. To further optimize the trade-off between the efficiency and the restoration accuracy, we propose a novel pruning method that determines the pruning ratio for N:M structured sparsity at each layer. Extensive experimental results on super-resolution and deblurring tasks demonstrate the efficacy of our method which outperforms previous pruning methods significantly. PyTorch implementation for the proposed methods will be publicly available at https://github.com/JungHunOh/SLS_CVPR2022

1. Introduction

Advances in deep learning has brought success in image restoration tasks such as super-resolution [29, 48] and deblurring [39, 60, 61]. Due to the heavy computational burden required by such methods, however, computing high-resolution images in practical applications has been challenging. Network pruning is one of the most popular tools to alleviate the computational burden of neural networks by eliminating weights that are less critical to the accuracy. It has shown remarkable efficacy in finding submodels for a better trade-off between accuracy and efficiency for image classification [9, 15, 17, 30, 34] and segmentation [14, 57].

Unstructured pruning [11, 13, 26] aims to find and remove individual weights that have relatively less impact on model accuracy. Still, accelerating the resulting models is difficult due to irregular sparsity patterns of weight tensors,

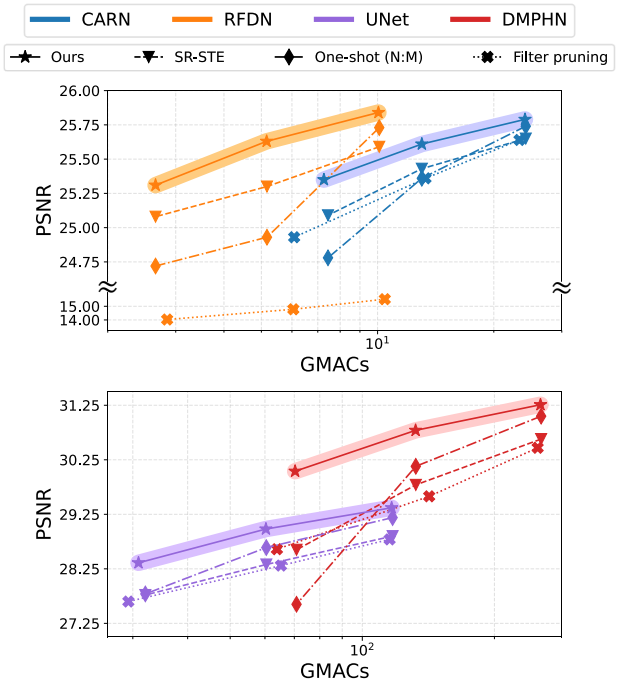


Figure 1. Trade-off between image restoration performance (PSNR) vs computational costs (MACs) on super-resolution (top) and deblurring tasks (bottom). We compare our method to the magnitude-based filter pruning [28] and the existing methods on N:M sparsity (One-shot pruning [37] and SR-STE [64]).

considering the complex nature of parallelization on GPUs. On the other hand, structured pruning removes predetermined structures (*e.g.*, a filter in convolution layers [28] or a channel of feature maps [17]) to enable the acceleration of pruned networks on GPUs. However, we empirically find image restoration models to be often susceptible to substantial performance degradation from structured pruning.

Recently, N:M fine-grained structured sparsity [19, 37, 64] has emerged as a better alternative, combining the strengths of both pruning methods: namely, fine-grained sparsity from unstructured pruning and hardware acceleration-ability from structured pruning. N:M structured sparsity enforces N number of weights in each group of M number of consecutive weights to have non-zero val-

ues. Such sparsity constraint has the potential of hardware acceleration, where 2:4 sparsity pattern has recently been supported in NVIDIA Ampere generation GPUs [37]. However, training a network with N:M sparsity has been proven to be difficult, which the existing works on N:M sparsity have focused on improving [37, 64]. Such difficulty has prevented the direct application of the already developed pruning techniques, such as pruning with layer-wise varying pruning ratios [27, 34], which is known to be crucial to the performance of the pruned networks. Particularly, we observed that several layers (*e.g.* the last upsampling layer) in image restoration networks are very sensitive to pruning with respect to the performance.

Here, we propose a layer-wise N:M sparsity search framework for efficient image restoration networks, named **S**earching for **L**ayer-wise N:M structured **S**parsity (SLS). In the prior arts [9, 17, 30, 31, 33], a filter or a channel is used as a unit of pruning but it is challenging to define the pruning unit in the case of N:M sparsity. To this end, we propose to consider the original weight tensor as the sum of sparse tensors whose configurations are determined by the magnitude of weights. We use each of sparse tensor as the unit of pruning in our N:M sparsity search problem. To learn how many units to preserve, we propose a trainable score for each pruning unit, which is designed to ensure units with the lowest magnitude-based importance are removed first for better performance.

Furthermore, since image restoration tasks often have different computational constraints, we present an adaptive inference method that uses several models trained by SLS with different efficiency. The proposed adaptive inference technique determines which pruned model should be used at inference time depending on the restoration difficulty of an input image patch. The adaptive inference method further improves the efficiency-accuracy trade-off and enables a flexible adoption of the computational budgets.

We summarize our contributions as follows:

- Observing the pruning sensitivity of each layer to be different, we propose a novel method, SLS, to determine the layer-wise N:M sparsity levels.
- From the mixture of the pruned models with different computational costs and accuracy, we propose to find a better trade-off with additional controllability at inference time.
- By extensive experiments with super-resolution and deblurring, we empirically validate our pruned models generally achieve state-of-the-art performance.

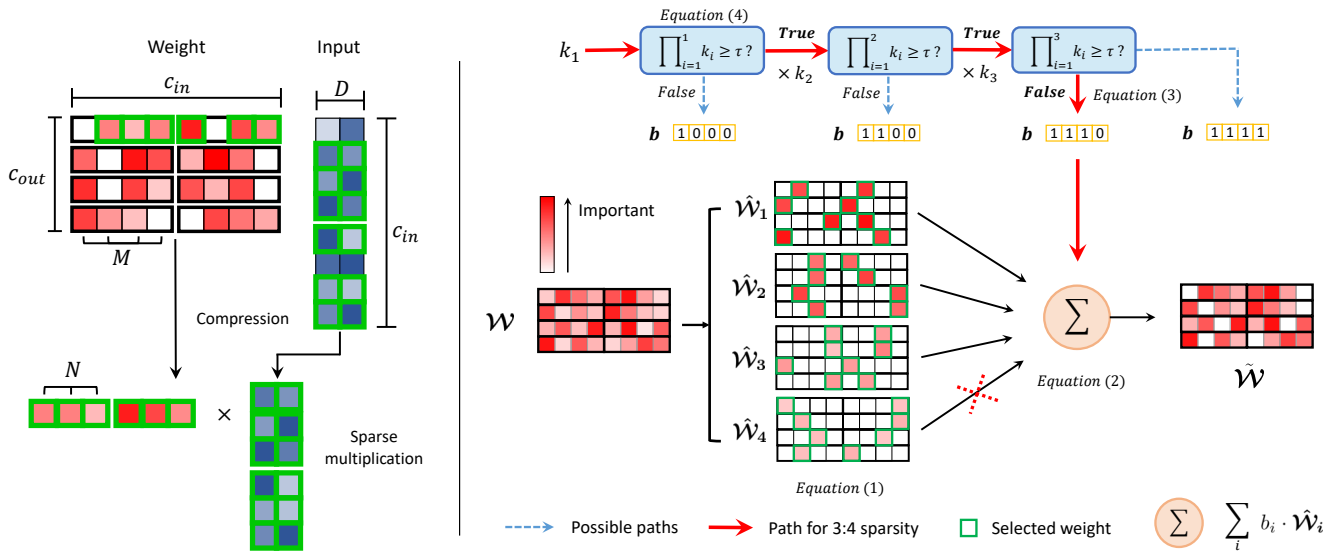
2. Related Work

Image Restoration. Motion blur or low resolution are common artifacts in images and restoring high-quality from such low-quality inputs has been widely studied in computer vision. In deep learning literature, many neural network architectures have been proposed to mitigate the artifacts. In image deblurring, multi-scale architectures [7, 39, 52], stacked networks [50, 58, 60], recurrent models [61] were proposed primarily to achieve better restoration quality. Due to the difficulty of ill-posed problem, such methods have employed complex architectures with high model capacity, leading to slow execution speed. Similarly, the advances in deep super-resolution from pioneering SRCNN [5] were made by studying various network architectures such as deep networks [23, 29], attention mechanisms [4, 42, 62], and dense connections [12, 53, 63].

In order to make deblurring and super-resolution fast, many efforts have been made to design light-weight architectures. For deblurring, [25] adopted Inception-ResNet-v2 [51] and MobileNetV2 [46] to build feature pyramid networks. Also, [44] used a shallow recurrent model and progressively deblurred an image. Similarly, for super-resolution, [2] and [32] proposed to use an effective convolutional module for an efficient yet accurate network. Furthermore, instead of designing models manually, neural architecture search (NAS) was adopted [3, 22, 49] to find efficient model structures. Different from the previous methods designing light-weight architectures, our method makes the existing models efficient by using N:M sparsity.

Network Pruning. From the early studies in neural networks, the redundant weights that have negligible impact on the final output have been witnessed and pruning aims to remove such unnecessary components [13, 26]. In unstructured pruning [6, 10, 11, 45], unimportant individual weight connects were eliminated, with the primary focus on accuracy preservation. However, due to the irregular sparsity patterns, accelerating the pruned networks requires specially designed hardware, limiting the application in practice [54]. On the other hand, structured pruning removes groups of the weights (*e.g.,* layers, filters, and channels of a feature map) in the network architecture, leading to easier acceleration on off-the-shelf devices. Especially, filter or channel pruning methods [9, 15–17, 27, 28, 30, 33, 35, 38, 56] have risen as a popular pruning strategy. They either train networks to find optimally pruned networks [9, 15, 27, 33, 56] or propose metrics to measure the relative importance of filters or channels in pre-trained models [16, 17, 28, 30, 35, 38, 43]. However, we observed that such coarse-grained structured sparsity can lead to significant damage to the performance in image restoration networks.

Recently, several approaches have been proposed in order to make the structured pruning at a more fine-grained level. Block-level sparsity with matrix math pipelines [8] is successfully accelerated with the known pruning structure. However, it requires to increase the feature size to maintain the original accuracy. A more fine-grained balanced sparsity is proposed in [55] which can be accelerated by grouping weights and pruning each group with a uni-



(a) Visual illustration for N:M sparsity

(b) Overview of the proposed method

Figure 2. (a) Visual illustration for N:M sparsity [37], where $N = 3$ and $M = 4$. We illustrate the sparse multiplication process only for the first row of the weight, where the non-zero weights and the input features at the corresponding positions are highlighted in green boxes. D refers to the spatial dimension of the input feature. (*e.g.*, input height × input width in a convolutional layer.) (b) Overview of the proposed method, SLS, which decomposes the weights into M ($M = 4$ in this case) groups based on the weight magnitude (Equation (1)). According to Equation (2), the final pruned weights are constructed based on the binary mask $\mathbf{b}$, each value of which controls whether the corresponding group will be used. The binary mask will be generated based on the priority of each importance group, following the operations outlined by Equation (3) and (4). We illustrate the case when the searched sparsity is 3:4.

form sparsity. Unlike the coarse-grained structured pruning methods, a pruned model by this method can achieve relatively higher accuracy by approximating feature map channels with the remaining weights. Recently, a similar idea was proposed as 2:4 fine-grained structured sparsity with hardware support on NVIDIA Ampere GPUs [37] and more general N:M sparsity configurations were explored in [19, 64]. However, the previous works only consider the uniform N:M sparsity levels over all layers, ignoring the layer-wise varying computational costs and contribution to the performance. In this paper, we further optimize the trade-off between the computational costs and performance by searching for an appropriate N:M sparsity level for each layer, specifically effective for extremely pruned networks.

3. Proposed Method

3.1. N:M sparsity

A weight tensor with N:M sparsity indicates a type of a weight tensor that satisfies the following conditions (See Figure 2a): (1) The number of input channels is divisible by M. (2) Each group of M consecutive weights should have at least N non-zero weights. With a weight satisfying the above constraints, the weight and input tensor are compressed by ignoring the zero weights and the corresponding input feature values. Then, the computations of the tensor multiplication between the compressed weight and input tensor are reduced to $\frac{N}{M}$ of the original computations.

3.2. Overview

The conventional structured pruning methods [9, 17, 30, 31, 33] remove the predetermined structural units (*e.g.*, filters or channels of a feature map). The common approach for finding layer-wise pruning ratios in structured pruning methods is to train a score value defined in each pruning unit and remove units with a small score value [9, 21, 33]. In the case of N:M sparsity pattern, however, it is challenging to determine such structural units because there are many possible configurations for preserving N weights out of M weights. To overcome the challenge, in Section 3.3, we consider the original weight tensor as the sum of M tensors with 1:M sparsity whose configurations of remaining weights are determined by the magnitude of weight. We propose to use each sparse tensor as the unit of pruning. Then, we propose a differentiable sparsity search framework that learns score values for each pruning unit by using a straight-through estimator [20]. We empirically found that the magnitude-based importance and the score value can lead to a conflict with respect to the importance rank of a pruning unit, which brings a substantial performance loss. In Section 3.4, we eliminate the conflict by ensuring the score value is aligned to the magnitude-based importance. In Section 3.5, we present our loss function and propose a loss annealing strategy to control the speed of pruning during training. In Section 3.6, we propose an adaptive inference method to improve the efficiency-accuracy trade-off.

3.3. Differentiable N:M Sparsity Search

Let $\mathcal{W}^l \in \mathbb{R}^{c^l_{out} \times c^l_{in} \times k^l_h \times k^l_w}$ denote the weight tensor in l-th convolutional layer. For notation simplicity, l is omitted unless otherwise noted. Our goal is to find the effective sparsity level of each layer given the target computational budgets. To this end, we represent the weight tensor as the sum of sparse tensors with 1:M sparsity:

$$\mathcal{W} = \sum_{i=1}^{M} \hat{\mathcal{W}}_i, \tag{1}$$

where $\hat{\mathcal{W}}_i$ is a weight tensor with 1:M sparsity in which only the i-th important weight is remained every M consecutive weights. We define the importance of each weight as its magnitude, which has been widely used in the pruning literature [19, 64]. By doing so, we can view the sparse tensor as a pruning unit in our problem setting. In other words, $\hat{\mathcal{W}}_i$ is analogous to a filter or a channel in the existing structured pruning methods. Then, we formulate the pruning ratio search problem as follows:

$$\tilde{\mathcal{W}} = \sum_{i=1}^{M} b_i \cdot \hat{\mathcal{W}}_i, \tag{2}$$

where b_i is a binary value indicating whether $\hat{\mathcal{W}}_i$ should be removed or not. To optimize b_i using gradient descent, we adopt straight-through estimator (STE) [20]:

$$b_i = \begin{cases} S(p_i, \tau) & \text{in a forward path,} \\ p_i & \text{in a backward path,} \end{cases} \tag{3}$$

where $S(\cdot, \tau)$ is a function that returns 1 for the greater value than a threshold τ and 0 otherwise and p_i is a priority score of $\hat{\mathcal{W}}_i$ that is learned during training.

3.4. Priority-Ordered Pruning

In section 3.3, we introduce the two importance measures for each sparse tensor: the magnitude of weight is for the definition of the pruning unit and the priority score is for learning the pruning ratio. However, the importance rank indicated by the two measures can be different, by which a pruning unit with weights of larger magnitude can be removed first before one with weights of smaller magnitude. We found such misalignment leads to substantial performance degradation. To solve this problem, we propose Priority-Ordered Pruning (POP) method that aligns the two importance measures by design. Specifically, we define the priority score as follows:

$$p_i \equiv \prod_{n=1}^{i-1} k_n \tag{4}$$

where $p_1 \equiv 1$ and k_n is a trainable parameter that is initialized to 1 and clamped to ensure $0 \leq k_n \leq 1$. By Equation (4), $p_{i+1} \leq p_i$ is guaranteed, so a pruning unit with weights of smaller magnitude is removed first. Figure 2b illustrates the overall process of the proposed learning framework.

3.5. Loss Function

For practical usage of the proposed method, we design our pruning framework as a budgeted pruning [9, 41], in which a network is pruned to meet the desired target computational budget. To this end, we first define the computational costs of a convolutional layer with N:M sparsity with respect to multiply-accumulate operations (MACs). Theoretically, the MACs of a convolutional layer with N:M sparsity are $\frac{N}{M}$ of the original ones. Thus, the computational costs of a pruned convolutional layer are defined as follows:

$$C_{pruned} = C_{original} \times \frac{\sum_{i=1}^{M} b_i}{M}, \tag{5}$$

where $C_{original} = (c_{out} \cdot c_{in} \cdot k_h \cdot k_w) \times (H \cdot W)$ denotes the original computational costs of the layer and H and W are the spatial sizes of the output of the layer. Then, we formulate our loss function as follows:

$$\mathcal{L} = \mathcal{L}_{task} + \lambda_{reg} \sum_{l=1}^{L} C^l_{pruned}, \tag{6}$$

where L is the total number of layers to be pruned, $\mathcal{L}_{task}$ is the task-specific loss function (*e.g.*, L1 loss for image restoration tasks) and $\sum_{l=1}^{L} C^l_{pruned}$ is the computational regularization loss. The two loss terms are balanced by the hyper-parameter λ_{reg}. Note that the gradient from the computational regularization loss can flow to k_i by using STE in Equation (3). Starting from a pretrained network, the network is pruned until satisfying $\sum_{l=1}^{L} C^l_{pruned} \leq C_{target}$, where C_{target} denotes a target computational budget. After reaching the target budget, all k_i are frozen and the pruned network is fine-tuned by optimizing the task-specific loss.

We empirically found that when the target computational budget is extremely low (*e.g.*, $C_{target} = 0.1 \times \sum_{l=1}^{L} C^l_{original}$), λ_{reg} should be large enough to reach the target budget. However, a large λ_{reg} can result in an aggressive performance degradation because the network is pruned too fast, which is hard to be recovered even after a fine-tuning process. To solve this problem, we set λ_{reg} as a small value at the beginning of training and gradually increase it according to the pruned rate change.At every predetermined K epoch, we measure the pruned rate change during the last K epochs and update λ_{reg} by following rules:

$$\lambda_{reg} = \begin{cases} \alpha \times \lambda_{reg} & \text{if } \Delta \frac{C_{pruned}}{C_{original}} \leq T, \\ \lambda_{reg} & \text{else,} \end{cases} \tag{7}$$

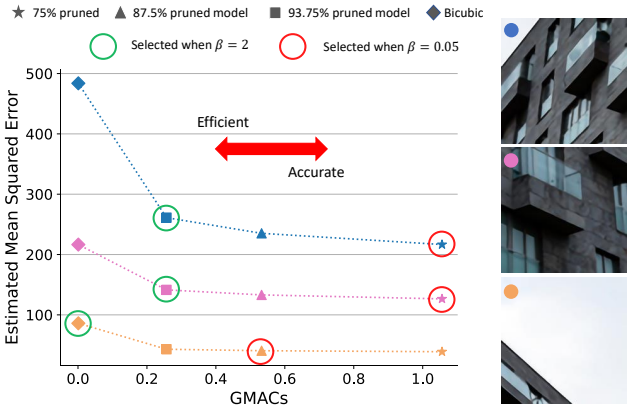


Figure 3. Visualization of the proposed adaptive inference method. Given the 3 images on the right side of the figure and 4 model candidates (75%, 87.5%, 93.75% pruned CARN [2] and bicubic upsampler), the trained MSE estimators estimate MSE between the restored image and the ground truth. Then, our method selects one model by Equation (8). By adjusting β, one can determine whether to focus on efficiency or accuracy.

where α is a hyper-parameter determining how fast λ_{reg} is increased and T is the threshold value of pruned rate change for updating λ_{reg}. Since the performance of the pruned network is not sensitive to the annealing hyper-parameters (α, T, and K), we fix them for all experiments.

3.6. Adaptive Inference

Each region in a low-quality image often has different restoration difficulties. For example, in the case of image super-resolution tasks, flat areas such as the sky can be easily restored by using only a few computational resources. Also, in order to restore large images (*e.g.*, 2K or 4K), it can be inevitable to process the whole image by dividing it into small patches due to the resource constraints [24]. From this motivation, we propose an adaptive inference method that determines which pruned models, trained by SLS, to use according to the restoration difficulty of an input image patch at inference time.

To quantify the restoration difficulty of a patch, we assume that the more an image patch is hard to be restored, the larger error between the ground truth and the restored result. Since the ground truth is not available at inference time, we use a light-weight convolutional neural network that can estimate the mean squared error (MSE) between the ground truth and the restored result from a target model. Given several models trained by SLS with different target computational budgets, we train the MSE estimators for each candidate model using training datasets.

Formally, given an image patch $\boldsymbol{x}$, our adaptive inference method selects a model among the candidates that is obtained by the following operation:

$$\underset{i}{\operatorname{argmax}} \frac{C_1 - C_i}{C_1} \times \beta + \frac{f_n(\boldsymbol{x}) - f_i(\boldsymbol{x})}{f_n(\boldsymbol{x})}, \tag{8}$$

where C_i and $f_i(\cdot)$ indicate the computational costs and the estimated MSE with respect to the i-th candidate model and n is the total number of candidates. We sort the index of each model with respect to the computational costs ($C_{i+1} \leq C_i$). By maximizing the two terms in Equation (8), our method tries to find the most efficient yet accurate model given $\boldsymbol{x}$. The hyper-parameter β enables flexible control of the computational costs of the selected model by making focus on either the computational costs or the performance. Figure 3 visualizes the proposed adaptive inference method.

4. Experiments

4.1. Dataset and Models

To validate the effectiveness of the proposed methods, we conduct experiments on image deblurring and super-resolution tasks. For image deblurring, we perform pruning on 3 model architectures that vary by the computational cost and the restoration accuracy: residual UNet [40], SRN [52] and DMPHN [60]. GOPRO dataset [39] is employed to train and evaluate the deblurring models. For image super-resolution, we use 3 popular and efficient architectures: EDSR [29], CARN [2] and RFDN [32]. We use DIV2K dataset [1] for training, Set14 [59], B100 [36] and Urban100 [18] benchmark datasets for evaluation.

4.2. Implementation Details

To make a fair comparison between different pruning methods, we train the networks with the same amount of iterations. The total training epochs are 4000 and 600, respectively for deblurring and super-resolution. Since the methods except for SR-STE [64] require a pretraining phase before pruning, we allocate half of the training epochs for pretraining and the rest for the pruning process for those methods. We set the hyper-parameters as $\tau = 0.5$, $\alpha = 1.1$, $T = 0.1$. λ_{reg} is set to 10^{-12} and 10^{-10} for image deblurring and super-resolution tasks, respectively. Also, we set $M = 32$ to allow extreme pruning ratios. For more details, please refer to the supplementary material.

4.3. Quantitative Comparison

In Table 1 and 2, we present the computational costs (GMACs) and the image restoration performance of the pruned models on image deblurring and super-resolution tasks, respectively. For each model, we train them using different computational budgets, 1/4, 1/8, and 1/16 of the original costs. Our method, SLS, is compared with the existing

Table 1. Image deblurring performance comparisons on GOPRO dataset [39].

Model	Method	GMACs	Num. Param.	PSNR$_\uparrow$ / SSIM$_\uparrow$ / LPIPS$_\downarrow$
UNet	Unpruned	458.04	6.79M	29.46 / 0.8837 / 0.1686
	One-shot (2:4)	230.84	3.40M	29.55 / 0.8849 / 0.1662
	Filter pruning	115.42	1.70M	28.79 / 0.8692 / 0.1893
	One-shot (8:32)	117.24	1.70M	29.19 / 0.8771 / 0.1795
	SR-STE (8:32)	117.24	1.70M	28.85 / 0.8691 / 0.1860
	SLS (Ours)	116.64	1.55M	**29.37 / 0.8811 / 0.1740**
	Filter pruning	65.27	956.74K	28.31 / 0.8570 / 0.2070
	One-shot (4:32)	60.44	851.38K	28.64 / 0.8646 / 0.1985
	SR-STE (4:32)	60.44	851.38K	28.33 / 0.8571 / 0.2070
	SLS (Ours)	60.31	797.30K	**28.98 / 0.8726 / 0.1870**
	Filter pruning	29.31	425.86K	27.65 / 0.8398 / 0.2325
	One-shot (2:32)	32.04	427.19K	27.79 / 0.8430 / 0.2345
	SR-STE (2:32)	32.04	427.19K	27.77 / 0.8431 / 0.2271
	SLS (Ours)	30.91	397.55K	**28.36 / 0.8573 / 0.2117**
SRN	Unpruned	1200.51	7.09M	30.28 / 0.9021 / 0.1310
	One-shot (2:4)	605.35	3.55M	30.53 / 0.9065 / 0.1264
	Filter pruning	302.67	1.77M	29.76 / 0.8915 / 0.1467
	One-shot (8:32)	307.77	1.78M	30.34 / 0.9030 / 0.1313
	SR-STE (8:32)	307.77	1.78M	29.91 / 0.8942 / 0.1407
	SLS (Ours)	306.84	1.72M	**30.45 / 0.9051 / 0.1283**
	Filter pruning	171.21	998.98K	29.33 / 0.8821 / 0.1603
	One-shot (4:32)	158.98	895.00K	29.87 / 0.8944 / 0.1427
	SR-STE (4:32)	158.98	895.00K	29.35 / 0.8839 / 0.1545
	SLS (Ours)	157.41	897.97K	**30.21 / 0.9006 / 0.1351**
	Filter pruning	76.94	444.87K	28.70 / 0.8691 / 0.1811
	One-shot (2:32)	84.59	252.50K	29.04 / 0.8766 / 0.1688
	SR-STE (2:32)	84.59	252.50K	28.83 / 0.8723 / 0.1747
	SLS (Ours)	84.52	489.81K	**29.75 / 0.8918 / 0.1470**
DMPHN	Unpruned	994.48	8.05M	31.22 / 0.9164 / 0.1243
	One-shot (2:4)	501.86	4.03M	31.43 / 0.9196 / 0.1192
	Filter pruning	250.94	2.02M	30.47 / 0.9043 / 0.1417
	One-shot (8:32)	255.55	2.02M	31.05 / 0.9137 / 0.1292
	SR-STE (8:32)	255.55	2.02M	30.63 / 0.9058 / 0.1406
	SLS (Ours)	254.64	2.24M	**31.26 / 0.9170 / 0.1242**
	Filter pruning	142.02	1.14M	29.58 / 0.8872 / 0.1639
	One-shot (4:32)	132.40	1.01M	30.13 / 0.8984 / 0.1504
	SR-STE (4:32)	132.40	1.01M	29.79 / 0.8916 / 0.1559
	SLS (Ours)	132.26	1.20M	**30.79 / 0.9097 / 0.1343**
	Filter pruning	63.90	506.41K	28.61 / 0.8662 / 0.1908
	One-shot (2:32)	70.82	506.88K	27.60 / 0.8393 / 0.2348
	SR-STE (2:32)	70.82	506.88K	28.60 / 0.8678 / 0.1831
	SLS (Ours)	70.29	603.47K	**30.04 / 0.8967 / 0.1525**

Table 2. Image super-resolution performace (PSNR$_\uparrow$) comparisons on benchmark datasets with the scaling factor of 4.

Model	Method	GMACs	Num. Param.	Set14 / B100 / Urban100
EDSR	Unpruned	114.49	1.52M	28.58 / 27.56 / 26.04
	One-shot (2:4)	58.22	765.00K	28.56 / 27.55 / 26.01
	Filter pruning	29.11	380.93K	28.44 / 27.48 / 25.75
	One-shot (8:32)	30.09	345.50K	**28.49** / 27.50 / 25.83
	SR-STE (8:32)	30.09	345.50K	28.44 / 27.46 / 25.73
	SLS (Ours)	29.56	363.39K	**28.49 / 27.51 / 25.84**
	Filter pruning	16.56	214.85K	28.35 / 27.41 / 25.59
	One-shot (4:32)	16.02	174.75K	28.34 / 27.41 / 25.52
	SR-STE (4:32)	16.02	174.75K	28.33 / 27.39 / 25.51
	SLS (Ours)	15.62	190.59K	**28.38 / 27.43 / 25.63**
	Filter pruning	7.52	96.00K	28.13 / 27.20 / 25.17
	One-shot (2:32)	8.98	89.38K	28.10 / 27.23 / 25.11
	SR-STE (2:32)	8.98	89.38K	28.13 / 27.27 / 25.22
	SLS (Ours)	8.65	97.28K	**28.22 / 27.32 / 25.31**
CARN	Unpruned	91.22	1.11M	28.49 / 27.49 / 25.82
	One-shot (2:4)	46.63	565.00K	28.49 / 27.51 / 25.86
	Filter pruning	23.31	279.46K	28.37 / 27.43 / 25.64
	One-shot (8:32)	24.20	278.76K	28.44 / 27.47 / 25.74
	SR-STE (8:32)	24.20	278.76K	28.40 / 27.42 / 25.65
	SLS (Ours)	24.09	276.34K	**28.46 / 27.48 / 25.79**
	Filter pruning	13.31	157.75K	28.26 / 27.32 / 25.36
	One-shot (4:32)	13.02	140.24K	28.24 / 27.32 / 25.36
	SR-STE (4:32)	13.02	140.24K	28.23 / 27.33 / 25.43
	SLS (Ours)	13.02	140.27K	**28.39 / 27.41 / 25.61**
	Filter pruning	6.08	70.61K	27.98 / 27.15 / 24.93
	One-shot (2:32)	7.44	70.97K	27.86 / 27.08 / 24.78
	SR-STE (2:32)	7.44	70.97K	28.03 / 27.19 / 25.09
	SLS (Ours)	7.26	67.18K	**28.22 / 27.32 / 25.35**
RFDN	Unpruned	39.86	828.75K	28.52 / 27.51 / 25.91
	One-shot (2:4)	20.02	416.25K	28.54 / 27.51 / 25.92
	Filter pruning	10.44	214.77K	16.50 / 17.32 / 15.52
	One-shot (8:32)	10.10	210.00K	28.46 / 27.47 / 25.73
	SR-STE (8:32)	10.10	210.00K	28.33 / 27.41 / 25.59
	SLS (Ours)	10.05	240.17K	**28.50 / 27.50 / 25.84**
	Filter pruning	6.05	123.84K	15.76 / 16.61 / 14.78
	One-shot (4:32)	5.17	106.88K	27.99 / 27.16 / 24.93
	SR-STE (4:32)	5.17	106.88K	28.19 / 27.31 / 25.30
	SLS (Ours)	5.16	139.60K	**28.41 / 27.43 / 25.63**
	Filter pruning	2.85	57.74K	15.01 / 15.82 / 14.03
	One-shot (2:32)	2.66	55.31K	27.49 / 27.06 / 24.72
	SR-STE (2:32)	2.66	55.31K	28.02 / 27.21 / 25.08
	SLS (Ours)	2.66	77.49K	**28.22 / 27.32 / 25.31**

pruning methods, filter pruning [28], one-shot N:M pruning [37] and SR-STE [64], with respect to PSNR, SSIM, and LPIPS. Under almost the same GMACs, SLS consistently achieves the best image restoration performance across all tasks and model architectures. Especially, SLS outperforms the other methods by a large margin at extremely pruned cases. These results show that our method can achieve a better trade-off between the computational costs and restoration performance by searching for the effective layer-wise N:M sparsity levels.

4.4. Qualitative Comparison

We present the qualitative results in Figure 5. In the case of image super-resolution tasks (the first two rows), we observe that the models trained by SLS can restore more sharp and clear textures, compared to the results from the other methods. Notably, the results from the filter pruning suffer checkerboard artifacts since pruning the last pixel shuffle upscaling layer [47] results in sparse pixel values. Similarly, the results on image deblurring tasks (the last two rows) show that the models trained by SLS can restore the detailed textures and cleaner car plates with better readability while other pruning methods fail to reconstruct such high-frequency details. The overall results show that under the same computational budgets, models pruned by SLS achieve perceptually satisfying performance.

4.5. Finding Optimal Pruning Ratio for Each Layer

Different from the previous methods that set a uniform N:M sparsity in all layers, our SLS finds the sparsity level for each layer via learning. In Figure 6, we visualize the searched level of sparsity of each layer. The results show that the searched N have a tendency to decrease when the computational costs of the corresponding layer become heavy. Interestingly, we found that this tendency is not shown in the last upscaling layer in RFDN (indicated as the 50-th layer in the figures). We empirically found that the last upscaling layer has a significant impact on the

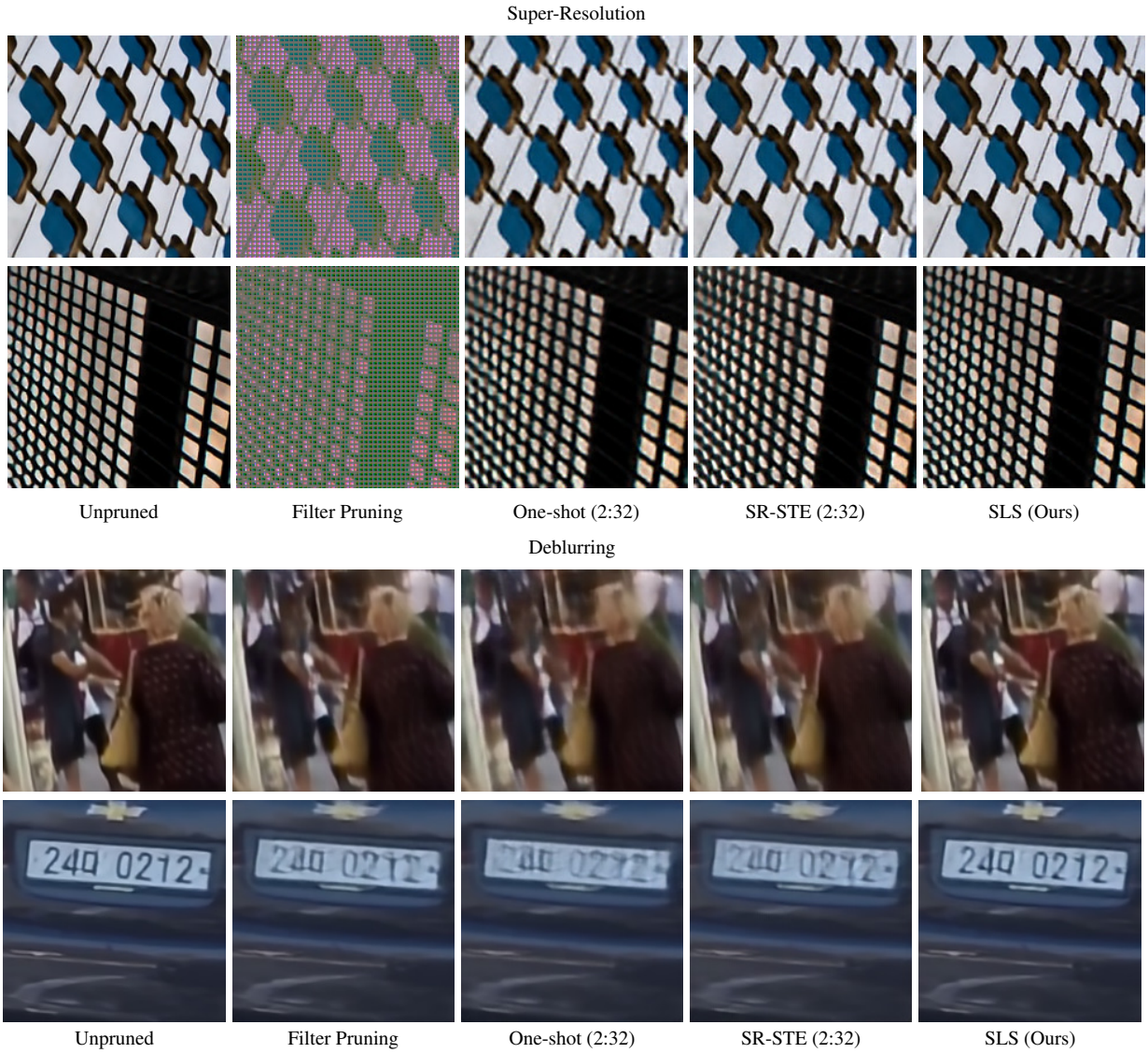


Figure 5. Qualitative comparisons with the existing pruning methods. The first two rows show image super-resolution results from RFDN with the scaling factor of 4. The last two rows show image deblurring results from DMPHN. For each task, all pruned models have almost the same computational costs (1/16 of the original value) with respect to GMACs.

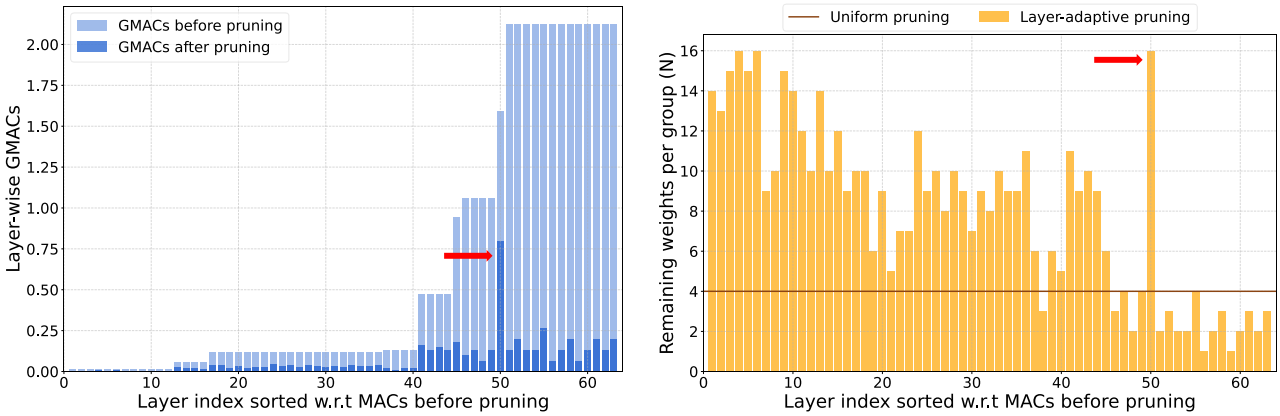
Table 3. Ablation studies on the proposed methods. In the 3rd column, the two numbers indicate the update periods (iterations) for updating $\hat{\mathcal{W}}_i^l$ in RFDN and UNet, respectively. For RFDN, we use Urban100 dataset. We set the target budget as $\frac{1}{8}$ of the original computational costs.

POP	λ_{reg} annealing	Update period (Iterations)	PSNR↑ RFDN	PSNR↑ UNet
✗	✓	1000/131	25.46	28.57
✓	✗	1000/131	25.59	28.84
✓	✓	No update	25.46	28.74
✓	✓	1/1	25.61	28.88
✓	✓	10000/1310	25.61	28.92
✓	✓	1000/131	**25.63**	**28.98**

performance since it is directly related to the final output. Thus, considering the better restoration performance in Table 2, these results demonstrate that SLS finds more effective pruning ratios for each layer by taking into account both the computational costs and the contribution to the performance of each layer.

4.6. Ablation Studies

To validate the effectiveness of each component in SLS, we conduct ablation studies and the results are shown in Table 3. To investigate the effect of Priority-Ordered Pruning (POP), we defined the priority score values p not as the cumulative product of auxiliary trainable parameters as in Equation (4) but as independent trainable parameters. The



(a) Visualization of the searched layer-wise computational costs (MACs)

(b) Visualization of the searched layer-wise N (when $M = 32$)

Figure 6. Analysis on the searched layer-wise N:M sparsity. The results are obtained by training RFDN model with $C_{target} = \frac{1}{8}C_{original}$. We visualize the layer-wise pruning ratios in terms of (a) GMACs and (b) N. The layer is sorted with respect to MACs before pruning. The highlighted bar by the red arrow indicates the result of the last upsampling layer in RFDN. The brown line in (b) is for the comparison with the uniform pruning [37, 64].

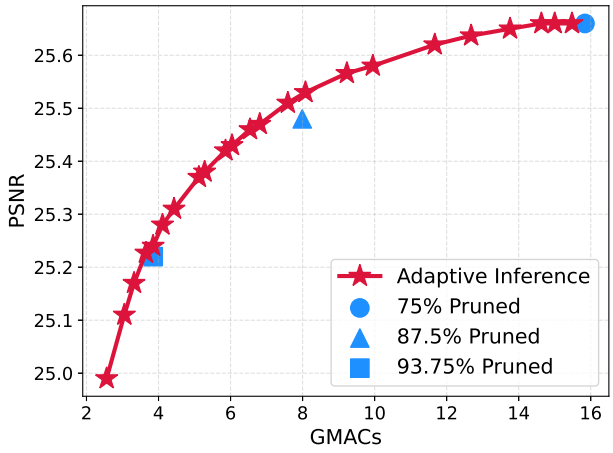


Figure 7. Results of the proposed adaptive inference method on Urban100 dataset with the scale factor of 4. We use the bicubic upsampler and three CARN models trained by SLS with different target budgets.

results show that there is a large performance loss when POP is not used, indicating that aligning the two importance measures (the magnitude of weights and the priority scores) is essential. For the ablation study on λ_{reg} annealing, we set λ_{reg} as a large value that is finally found when the annealing strategy is used and train the models with it. The results demonstrate that λ_{reg} annealing brings an additional performance gain by controlling the speed of pruning process according to the current pruned rate. In Equation (1), we group the weight tensor into several sparse tensors by using the magnitude of weights. Since the weights change during training, we should update the sparse tensors, $\hat{\mathcal{W}}_i$. To find an appropriate update period, we train models with different update periods. As expected, the pruned models suffer significant performance degradation when there is no update. Also, updating $\hat{\mathcal{W}}_i$ at every iteration is not helpful, so we set the update period as 10000 and 1310 iterations (10 epochs) for super-resolution and deblurring, respectively.

4.7. Adaptive Inference

Figure 7 shows the results of the proposed adaptive inference method on image super-resolution tasks. An input image is divided into several patches and each patch is restored by the selected model by Equation (8). As shown in the figure, our adaptive inference scheme not only enables the detailed control of computational budgets but also improves the trade-off between the computational costs and the restoration performance in terms of PSNR. For the experimental details and the results on deblurring tasks, please refer to the supplementary material.

5. Conclusion

In this paper, we propose a novel layer-wise pruning ratio search framework, SLS, tailored for N:M sparsity. Our differentiable learning framework is trained end-to-end with the task-specific and the computational regularization loss to determine a more effective degree of sparsity for each layer. Compared with the previous methods with uniform N:M sparsity at all layers, our results achieve state-of-the-art image restoration performance at similar computational budgets. Furthermore, our adaptive inference scheme facilitates the detailed control of the computational budgets with improved restoration performance.

Acknowledgment. This work was supported in part by IITP grant funded by the Korea government (MSIT) [No. 2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University), and No. 2021-0-02068, Artificial Intelligence Innovation Hub], and in part by AIRS Company in Hyundai Motor Company & Kia Motors Corporation through HMC/KIA-SNU AI Consortium.

References

[1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, 2017. 5

[2] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, 2018. 2, 5

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

[4] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *CVPR*, 2019. 2

[5] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE TPAMI*, 38(2):295–307, 2015. 2

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

[7] Hongyun Gao, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Dynamic scene deblurring with parameter selective sharing and nested skip connections. In *CVPR*, 2019. 2

[8] Scott Gray, Alec Radford, and Diederik P Kingma. Gpu kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224*, 3, 2017. 2

[9] Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: differentiable markov channel pruning for neural networks. In *CVPR*, 2020. 1, 2, 3, 4

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

[11] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 1, 2

[12] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *CVPR*, 2018. 2

[13] Babak Hassibi and David G Stork. Second order derivatives for network pruning: optimal brain surgeon. In *NIPS*, 1993. 1, 2

[14] Wei He, Meiqing Wu, Mingfu Liang, and Siew-Kei Lam. Cap: context-aware pruning for semantic segmentation. In *WACV*, 2021. 1

[15] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *CVPR*, 2020. 1, 2

[16] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*, 2019. 2

[17] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 1, 2, 3

[18] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 5

[19] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Seffi Naor, and Daniel Soudry. Accelerated sparse neural training: a provable and efficient method to find n:m transposable masks. In *NeurIPS*, 2021. 1, 3, 4

[20] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *NIPS*, 2016. 3, 4

[21] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *ICML*, 2020. 3

[22] Heewon Kim, Seokil Hong, Bohyung Han, Heesoo Myeong, and Kyoung Mu Lee. Fine-grained neural architecture search. *arXiv preprint arXiv:1911.07478*, 2019. 2

[23] Jiwon Kim, Jungkwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 2

[24] Xiangtao Kong, Hengyuan Zhao, Yu Qiao, and Chao Dong. Classsr: a general framework to accelerate super-resolution networks by data characteristic. In *CVPR*, 2021. 5

[25] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. DeblurGAN-v2: Deblurring (orders-of-magnitude) faster and better. In *ICCV*, 2019. 2

[26] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *NIPS*, 1990. 1, 2

[27] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: fast sub-net evaluation for efficient neural network pruning. In *ECCV*, 2020. 2

[28] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017. 1, 2, 6

[29] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 1, 2, 5

[30] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: filter pruning using high-rank feature map. In *CVPR*, 2020. 1, 2, 3

[31] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *CVPR*, 2019. 2, 3

[32] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *ECCV Workshops*, 2020. 2, 5

[33] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 2, 3

[34] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. Metapruning: meta learning for automatic neural network channel pruning. In *ICCV*, 2019. 1, 2

[35] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017. 2

[36] 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. 5

[37] Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021. 1, 2, 3, 6, 8

[38] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *CVPR*, 2019. 2

[39] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017. 1, 2, 5, 6

[40] Seungjun Nah, Sanghyun Son, Jaerin Lee, and Kyoung Mu Lee. Clean images are hard to reblur: Exploiting the ill-posed inverse task for dynamic scene deblurring. In *ICLR*, 2022. 5

[41] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang. Dsa: more efficient budgeted pruning via differentiable sparsity allocation. In *ECCV*, 2020. 4

[42] 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. 2

[43] Junghun Oh, Heewon Kim, Sungyong Baik, Cheeun Hong, and Kyoung Mu Lee. Batch normalization tells you which filter is important. In *WACV*, 2022. 2

[44] Dongwon Park, Dong Un Kang, Jisoo Kim, and Se Young Chun. Multi-temporal recurrent neural networks for progressive non-uniform single image deblurring with incremental temporal training. In *ECCV*, 2020. 2

[45] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *CVPR*, 2020. 2

[46] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: inverted residuals and linear bottlenecks. In *CVPR*, 2018. 2

[47] Wenzhe Shi, Jose Caballero, Ferenc Huszar, 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. 6

[48] Sanghyun Son and Kyoung Mu Lee. Srwarp: Generalized image super-resolution under arbitrary transformation. In *CVPR*, 2021. 1

[49] 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. 2

[50] Maitreya Suin, Kuldeep Purohit, and A. N. Rajagopalan. Spatially-attentive patch-hierarchical network for adaptive motion deblurring. In *CVPR*, 2020. 2

[51] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 2

[52] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *CVPR*, 2018. 2, 5

[53] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks. In *ECCV Workshops*, 2018. 2

[54] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NIPS*, 2016. 2

[55] Zhuliang Yao, Shijie Cao, Wencong Xiao, Chen Zhang, and Lanshun Nie. Balanced sparsity for efficient dnn inference on gpu. In *AAAI*, 2019. 2

[56] Jianbo Ye, Xin Lu, Zhe Lin, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *ICLR*, 2018. 2

[57] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: global filter pruning method for accelerating deep convolutional neural networks. In *NeurIPS*, 2019. 1

[58] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *CVPR*, 2021. 2

[59] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In Jean-Daniel Boissonnat, Patrick Chenin, Albert Cohen, Christian Gout, Tom Lyche, Marie-Laurence Mazure, and Larry Schumaker, editors, *Curves and Surfaces*, pages 711–730, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 5

[60] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for image deblurring. In *CVPR*, 2019. 1, 2, 5

[61] Jiawei Zhang, Jinshan Pan, Jimmy Ren, Yibing Song, Linchao Bao, Rynson W.H. Lau, and Ming-Hsuan Yang. Dynamic scene deblurring using spatially variant recurrent neural networks. In *CVPR*, 2018. 1, 2

[62] 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. 2

[63] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018. 2

[64] Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch. In *ICLR*, 2021. 1, 2, 3, 4, 5, 6, 8