Efficient Joint Optimization of Layer-Adaptive Weight Pruning in Deep Neural Networks

Kaixin Xu[1,2,*] Zhe Wang[1,2,*] Xue Geng[1] Min Wu[1] Xiaoli Li[1,2] Weisi Lin[2]
[1]Institute for Infocomm Research (I^{2}R), Agency for Science, Technology and Research (A*STAR),
1 Fusionopolis Way, 138632, Singapore
[2] Nanyang Technological University, Singapore
{xuk,wangz,geng_xue,wumin,xlli}@i2r.a-star.edu.sg, wslin@ntu.edu.sg

Abstract

In this paper, we propose a novel layer-adaptive weight-pruning approach for Deep Neural Networks (DNNs) that addresses the challenge of optimizing the output distortion minimization while adhering to a target pruning ratio constraint. Our approach takes into account the collective influence of all layers to design a layer-adaptive pruning scheme. We discover and utilize a very important additivity property of output distortion caused by pruning weights on multiple layers. This property enables us to formulate the pruning as a combinatorial optimization problem and efficiently solve it through dynamic programming. By decomposing the problem into sub-problems, we achieve linear time complexity, making our optimization algorithm fast and feasible to run on CPUs. Our extensive experiments demonstrate the superiority of our approach over existing methods on the ImageNet and CIFAR-10 datasets. On CIFAR-10, our method achieves remarkable improvements, outperforming others by up to 1.0% for ResNet-32, 0.5% for VGG-16, and 0.7% for DenseNet-121 in terms of top-1 accuracy. On ImageNet, we achieve up to 4.7% and 4.6% higher top-1 accuracy compared to other methods for VGG-16 and ResNet-50, respectively. These results highlight the effectiveness and practicality of our approach for enhancing DNN performance through layer-adaptive weight pruning. Code will be available on `https://github.com/Akimoto-Cris/RD_VIT_PRUNE`.

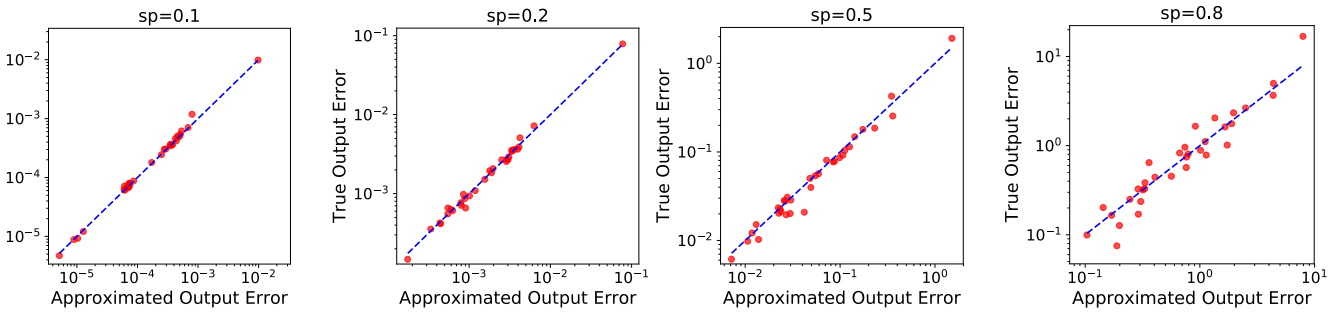
1. Introduction

Deep Neural Networks (DNNs) [22, 34, 35, 17, 19] play a critical role in various computer vision tasks. However, to achieve high accuracy, DNNs typically require large number of parameters, which makes it very energy-consuming and is difficult to be deployed on resource-limited mobile devices [16, 15]. Pruning is one of the powerful ways to reduce the complexity of DNNs. By removing the redundant parameters, the operations can be significantly reduced (e.g., FLOPs), which leads to faster speed and less energy-consuming. Typically, pruning approaches can be divided into two categories: structured pruning [14, 1, 9, 31, 18, 30] and weight (unstructured) pruning [27, 32, 28, 39, 16, 15]. Structured pruning approaches consider a channel or a kernel as a basic pruning unit, while weight pruning approaches consider a weight as a basic pruning unit. The former is more hardware-friendly and the latter is able to achieve higher pruning ratio.

In this paper, we focus on improving the weight pruning and propose a novel jointly-optimized layer-adaptive approach to achieve state-of-the-art results between FLOPs and accuracy. Recent discoveries [10, 13, 25] demonstrate that layer-adaptive sparsity is the superior pruning scheme. However, one drawback in prior layer-adaptive approaches is that they only consider the impact of a single layer when deciding the pruning ratio of that layer. The mutual impact between different layers is ignored. Moreover, another challenge is that the search space of the pruning ratio for each layer increases exponentially as the number of layers. In a deep neural network, the number of layers can be a hundred or even a thousand, which makes it very difficult to find the solution efficiently.

In our approach, we define a joint learning objective to learn the layer-adaptive pruning scheme. We aim to minimize the output distortion of the network when pruning weights on all layers under the constraint of target pruning ratio. As the output distortion is highly related to accuracy, our approach is able to maintain accuracy even at high pruning ratios. We explore an important property of the output distortion and find that the additivity property [42, 41, 38] holds when we prune weights on multiple layers. In other words, the output distortion caused by pruning all layers' weights equals to the sum of the output distortion due to the

[*]These authors contributed equally to this work



(a) Sparsity=0.1. (b) Sparsity=0.2. (c) Sparsity=0.5. (d) Sparsity=0.8.

Figure 1: An example of additivity property collected on ResNet-32 on CIFAR-10. The vertical axis shows the output distortion when pruning only two consecutive layers. The horizontal axis shows the sum of the output distortion due to the pruning the involved two layers individually. Sub-figures display the situations when all layers in the model are assigned with the corresponding sparsity.

pruning of each individual layer. We provide a mathematical derivation for the additivity property by using the Taylor series expansion.

Moreover, utilizing the additivity property, we develop a very fast method to solve the optimization via dynamic programming, which has only linear time complexity. We rewrite the objective function as a combinatorial optimization problem. By defining the state function and the recursive equation between different states, we can decompose the whole problem into sub-problems and solve it via dynamic programming. In practice, our approach is able to find the solution in a few minutes on CPUs for deep neural networks. Note that different from other approximation algorithms, dynamic programming is able to find the global optimal solution, which means that our approach provides optimal pruning scheme with minimal output distortion. We summarize the main contributions of our paper as follows:

- We propose a novel layer-adaptive pruning scheme that jointly minimizes the output distortion when pruning the weights in all layers. As the output distortion is highly related to the accuracy, our approach maintains high accuracy even when most of the weights are pruned. We also explore an important additivity property for the output distortion based on Taylor series expansion.
- We develop a fast algorithm to solve the optimization via dynamic programming. The key idea is to rewrite the objective function as a combinatorial optimization problem and then relax the whole problem into tractable sub-problems. Our method can find the solution of a deep neural network in a few minutes.
- Our approach improves state-of-the-arts on various deep neural networks and datasets.

The rest of our paper is organized as follows. We discuss the related works in section 2. In section 3, we develop our approach in detail. We present the objective function, the optimization method, and the time complexity analysis of the algorithm. In the last section, we provide the comprehensive experimental results.

2. Related Works

Our focus of this work generally falls into the magnitude-based pruning (MP) track within model compression of neural networks, with early works such as OBD [24]. MP is done by ranking or penalizing weights according to some criterion (*e.g.* magnitude) and removing low-ranked weights. Many efforts have been made ever since under the context of [24, 16], which can be roughly divided into the following approaches depending on their timing of pruning embedded in the network training.

Post-training Pruning. Post-training pruning scheme prunes out network parameters after standard network training, *i.e.* prunes from a pretrained converged model. Under this context, parameters can be pruned out at once to achieve target sparsity constraint (ont-shot pruning), or pruned out gradually during the sparse model fine-tuning (iterative pruning). [16] proposed an iterative pruning scheme that determines layerwise sparsity using layer statistics heuristic. [45, 10] adopted a global pruning threshold throughout all layers in the network to meet the model sparsity constraint. [5] [33] pooled all layers together and determined pruning thresholds for different layers in an integrated fashion. [12] proposed to rewind the weights from previous iterative pruning phase based on the lottery ticket hypothesis. LAMP[25] derived a closed-form layerwise sparsity selection from a relaxed layerwise $l2$ distortion minimization problem that is compatible with various post-training pruning schemes including iterative and one-shot pruning. PGMPF [4] adopted simple $l2$-based layerwise pruning criterion and improved the weight masking and

updating rules during finetuning. [6] adopted a one-shot pruning method by leveraging zero-invariant groups. [23] proposed to re-calibrate the biases and variances of model weights and activations, similar to the widely adopted bias correction in model quantization [11, 2]. [32] presented an iterative-pruning method that leverage taylor expansion of model loss and derived a gradient based pruning criteria. Our method leverages taylor expansion on output distortion parametrized by layer weights, which is fundamentally different from [32]. SuRP [20] recursively applies triangular inequality and assumes laplacian distribution to approximate output distortion to achieve joint-optimzation similar to us. However, our approximation is more straight-forward and do not need any assumptions on the distribution.

Pruning at Initialization. In contrast to the previous scheme, there is an emerging line of work that aims to remove connections or neurons from scratch at the initialization of training, with the merit of avoiding pretraining and complex pruning schedules. SNIP [26] prunes parameters only once at the initialization phase of training. The normalized magnitudes of the derivatives of parameters are defined as the pruning criterion. [7] presented a modified saliency metric based on SNIP [26], allowing for calculating saliences of partially pruned networks. [36] engineered the gradient flow when training sparse networks from scratch to achieve better convergence. Since pruning at initialization out of our research scope, one may refer to related surveys [37] for more comprehensive introduction.

Other Pruning Schemes. [3] interleaves the pruning in between normal training course, gradually pruning out more connections and neurons from the networks. This scheme is similar to the previous iterative pruning, however, here the model is trained from scratch. ProbMask [43] similarly leverages projected gradient descent with progressive pruning stretegy to directly train sparse networks. [40] integrates supermask training with gradient-drive sparsity for training sparse networks.

Since our main contribution is the improvement of the pruning criteria, we mainly evaluate our method under post-training unstructured pruning paradigms, such as iterative pruning and one-shot pruning. Although our method may have equal potential effectiveness on other sparsity structures and pruning schemes like Pruning at Initialization, we leave such validations for future works.

3. Approach

In this section, we present our approach in detail. We first give the formulation of our objective function and then provide the optimization method. An additivity property is derived based on Taylor series approximation. The implementation details of the dynamic programming and the analysis of the time complexity are also provided.

3.1. Objective Function

Following the notations in [25], let f denote a neural network, define $W^{(1:l)} = \left(W^{(1)}, ..., W^{(l)}\right)$ as all the parameters of f, where l is the number of layers and $W^{(i)}$ is the weights in layer i. When we prune part of the parameters of f, we will receive a modified neural network with the new parameter set $\tilde{W}^{(1:l)}$. We view the impact of pruning as the distance between the network outputs $f(x; W^{(1:l)})$ and $f(x; \tilde{W}^{(1:l)})$. The learning objective is to minimize the output distortion caused by pruning under the constraint of the pruning ratio,

$$\min \|f(x; W^{(1:l)}) - f(x; \tilde{W}^{(1:l)})\|^2 \quad s.t. \frac{\|\tilde{W}^{(1:l)}\|_0}{\|W^{(1:l)}\|_0} \leq R, \quad (1)$$

where R denotes the pruning ratio for the entire network.

An important property we discover is that the expectation of output distortion, caused by pruning all layers' weights, equals the sum of expectation of output distortion due to the pruning of each individual layer,

$$E\left(\|f(x; W^{(1:l)}) - f(x; \tilde{W}^{(1:l)})\|^2\right) = \sum_{i=1}^{l} E(\delta_i), \quad (2)$$

where δ_i denotes the output distortion when only pruning the weights in layer i.

3.2. Analysis

We provide a mathematical derivation for the additivity property. We make the following two assumptions for the proof of additivity property:

Assumption 1 ***Taylor first order expansion****: The neural network f parametrized by $W^{(1:l)}$ when given a small perturbation $\Delta W^{(1:l)}$ resulting in $\tilde{W}^{(1:l)} = W^{(1:l)} + \Delta W^{(1:l)}$ can be expanded as the following:*

$$f(x; \tilde{W}^{(1:l)}) = f(x; W^{(1:l)}) + \sum_{i=1}^{l} \frac{\partial f}{\partial W^{(i)}} \Delta W^{(i)}. \quad (3)$$

Assumption 2 ***I.d.d. weight perturbation across layers [44]:*** $\forall 0 < i \neq j < L, E(\Delta W^{(i)})E(\Delta W^{(j)}) = 0.$

According to Eq. (3), $\delta = \|f(x; W^{(1:l)}) - f(x; \tilde{W}^{(1:l)})\|^2$ can be written as

$$\delta = \Big(\sum_{i=1}^{l} \Delta W^{(i)^\top} \frac{\partial f}{\partial W^{(i)}}^\top\Big)\Big(\sum_{j=1}^{l} \frac{\partial f}{\partial W^{(j)}} \Delta W^{(j)}\Big). \quad (4)$$

When we take the expectation of Eq. (4) for both sides, the right hand side can be opened up into additive terms (vector transpose is agnostic inside expectation):

$$E(\delta) = \sum_{1 \leq i,j \leq l} E\left(\Delta W^{(i)} \frac{\partial f}{\partial W^{(i)}}\right) E\left(\Delta W^{(j)} \frac{\partial f}{\partial W^{(j)}}\right). \quad (5)$$

Further, since the derivative $\frac{\partial f}{\partial W^{(i)}}$ is a constant as we consider trained fixed network weights, we can derive the following from Assumption 2:

$$E\left(\Delta W^{(i)}\frac{\partial f}{\partial W^{(i)}}\right)E\left(\Delta W^{(j)}\frac{\partial f}{\partial W^{(j)}}\right)=0. \quad (6)$$

Therfore, the cross terms ($i \neq j$) in Eq. (5) disappear, obtaining:

$$E(\delta)=\sum_{i=1}^{l}E\left(\|\frac{\partial f}{\partial W^{(i)}}\Delta W^{(i)}\|^2\right). \quad (7)$$

Eq. (7) is the result we want because, again, according to Assumption 1,

$$\begin{aligned}\frac{\partial f}{\partial W^{(i)}}\Delta W^{(i)} = f(x;W^{(1:i-1)},\tilde{W}^{(i)},W^{(i+1,l)}) \\ - f(x;W^{(1;l)}).\end{aligned} \quad (8)$$

Therefore, the left hand side of Eq. (7) becomes the real output distortion δ when all layers are pruned, and the right hand side becomes the sum of the output distortion due to the individual pruning of each single layer's weights, which can be used to approximate the output distortion.

We have done an empirical examination of our theorectically proposed additivity property on real network. As shown in Fig. 1, when we examine the cases where only pruning two adjacent layers each time in a pretrained model, contributing to the right hand side addable distortion terms while other layers contributing zero to the approximation, we observe that the additivity holds quite well with marginal residuals, where almost all observation points sit close to the identity line.

3.3. Optimization via Dynamic Programming

By utilizing the additivity property, we can rewrite the objective function as a combinatorial optimization problem and solve it efficiently using dynamic programming. The objective function is re-written as,

$$\min\ \delta_1+\delta_2+...+\delta_l \quad s.t.\ \ t_1+t_2+...+t_l=T, \quad (9)$$

where T denotes the total number of weights to prune and t_i denotes the number of weights to prune in layer i. We solve (9) by decomposing the whole problem into sub-problems. The basic idea is that we define a state function and find the recursive equation between the states. The problem is solved based on the recursive equation.

Specifically, define g as the state function, in which g_i^j means the minimal distortion caused when pruning j weights at the first i layers. Our goal is to calculate g_l^T. For initialization, we have,

$$g_1^j=\delta_1(j),\ for\ \ 1\leq j\leq T, \quad (10)$$

Algorithm 1 Optimization via dynamic programming.

Input: Output distortion $\delta_i(j)$ when pruning j weights in single layer i, for $1\leq i\leq l$ and $1\leq j\leq T$.
Output: The number of weights p_i pruned in layer i.
Initialize minimal output distortion $g_i^j=0$ when pruning j weights in the first i layers.
Initialize state function $s_i^j=-1$ where s_i^j denotes the number of weights pruned in layer i when pruning j weights in the first i layers.
for i from 1 to l **do**
  for j from 0 to T **do**
    If $i=1$: $g_1^j=\delta_1(j)$, $s_1^j=j$.
    Else: $g_i^j=\min\{g_{i-1}^{j-k}+\delta_i(k)\}$, $s_i^j=\arg\min_k\{g_i^j\}$.
  end for
end for
The number of weights pruned in layer l is $p_l=s_l^T$.
Update $T=T-s_l^T$.
for i from $l-1$ to 1 **do**
  The number of weights pruned in layer i is $p_i=s_i^T$.
  Update $T=T-s_i^T$.
end for

where $\delta_i(j)$ denotes the distortion caused when pruning j weights at layer i. Then we have the recursive equation between the states g_i and g_{i-1}, which is,

$$g_i^j=\min\{g_{i-1}^{j-k}+\delta_i(k)\},\ where\ \ 1\leq k\leq j. \quad (11)$$

The state functions are calculated based on equation (11) in a bottom-up manner from g_1 to g_l. In practice, we need another variable s to store the decision of each state to know the number of weights pruned in each layer. s is defined as

$$s_i^j=\arg\min_k\{g_i^j=g_{i-1}^{j-k}+\delta_i(k)\}. \quad (12)$$

Algorithm 1 shows the pseudo-codes to calculate the state function and find the pruning solution.

3.4. Time complexity analysis

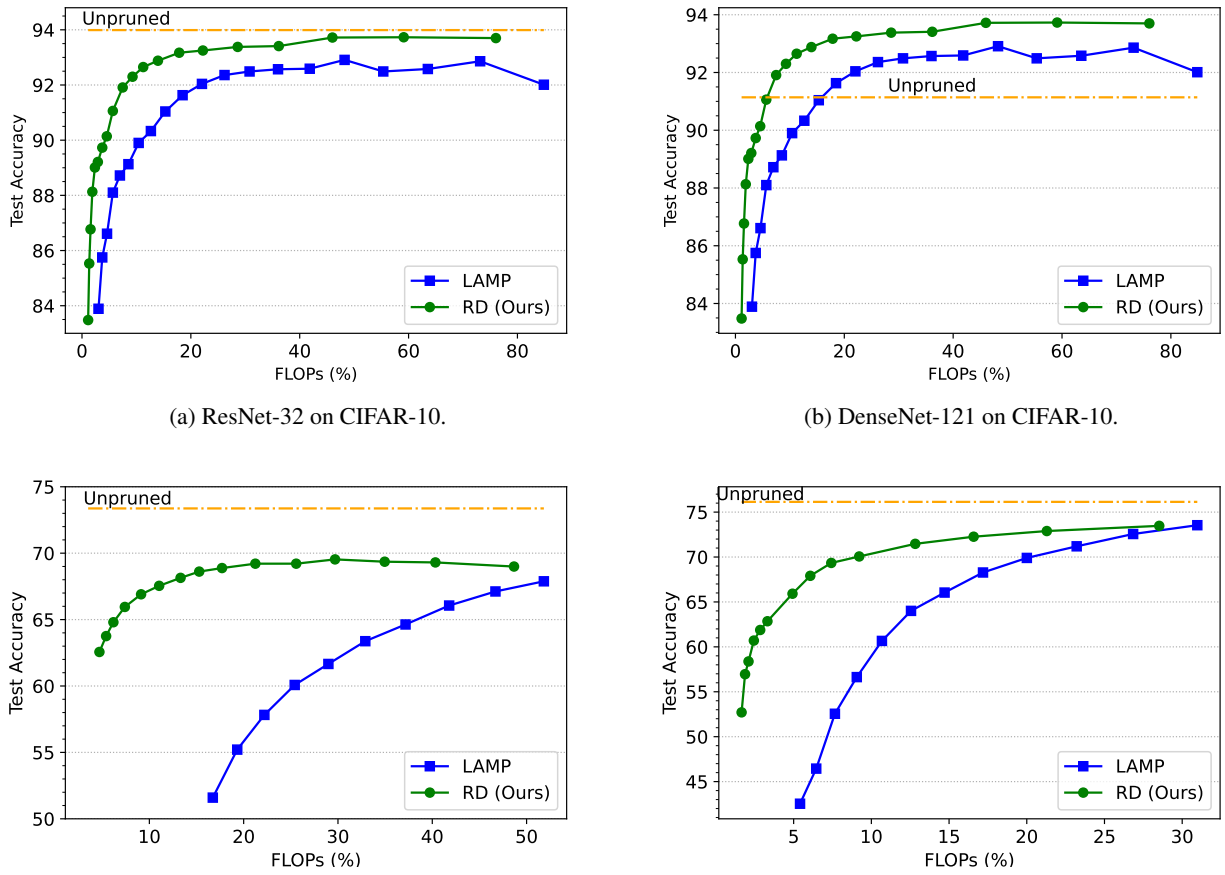
The time complexity of the optimization algorithm using dynamic programming is $O(l\times T^2)$, as we have $l\times T$ different states, and each state needs to enumerate the number of weights pruned in a layer. In practice, this algorithm is very fast which costs just a few seconds on CPUs for deep neural networks. We show the detailed results of the speed in the experimental section.

4. Experiment Results

Implementation Details. As our contribution to the existing pruning schemes is on the layer-wise sparsity selection, we evaluate our rate-distortion-based pruning method under different experimental settings, including iterative pruning and one-shot pruning, as well as on multiple network

Dataset	Arch	Method	Sparsity (%)	Remaining FLOPs (%)	Top-1 (%) ↑	Top-1 drop (%) ↓
CIFAR-10	ResNet-32 (Dense: 93.99)	LAMP [25]	79.03	36.02	92.58 ± 0.25	1.41
		Ours	58.65	34.5	$\mathbf{93.62 \pm 0.23}$	**0.37**
		LAMP [25]	89.3	21.66	91.94 ± 0.24	2.05
		Ours	73.76	22.21	$\mathbf{93.34 \pm 0.10}$	**0.65**
		LAMP [25]	95.5	11	90.04 ± 0.13	3.95
		Ours	86.57	11.25	$\mathbf{92.56 \pm 0.20}$	**1.43**
		LAMP [25]	98.85	3.29	83.66 ± 0.29	10.33
		Ours	95.5	3.59	$\mathbf{90.83 \pm 0.24}$	**3.16**
	VGG-16 (Dense: 91.71)	E-R ker. [10]	95.6	/	91.99 ± 0.14	-0.79
		DPF [29]	95	/	93.87 ± 0.15	-0.13
		LAMP [25]	95.6	15.34	92.06 ± 0.21	-0.86
		SuRP [20]	95.6	/	92.13	-0.93
		Ours	95.6	6.83	$\mathbf{92.59 \pm 0.17}$	$\mathbf{-0.88}$
		Global [33]	98.85	/	81.56 ± 3.73	9.64
		Uniform [45]	98.85	/	55.68 ± 12.20	35.52
		Uniform+ [13]	98.85	/	87.85 ± 0.26	3.35
		E-R ker. [10]	98.85	/	90.55 ± 0.19	0.65
		LAMP [25]	98.85	6.65	91.07 ± 0.4	0.13
		SuRP [20]	98.84	/	91.21	-0.01
		Ours	98.85	3.43	$\mathbf{92.14 \pm 0.18}$	$\mathbf{-0.43}$
		PGMPF [4]	/	33	**93.6**	0.08
		LAMP [25]	86.58	33.53	92.22 ± 0.05	-0.51
		Ours	67.21	35.49	92.76 ± 0.18	$\mathbf{-1.05}$
	DenseNet-121 (Dense: 91.14)	LAMP [25]	95.5	6.45	90.11 ± 0.13	1.03
		SuRP [20]	95.5	/	90.75	0.39
		Ours	95.5	6.72	$\mathbf{91.49 \pm 0.21}$	$\mathbf{-0.35}$
		Global [33]	98.85	/	45.30 ± 27.75	45.84
		Uniform [45]	98.85	/	66.46 ± 18.72	24.68
		Uniform+ [13]	98.85	/	69.25 ± 19.28	21.89
		E-R ker. [10]	98.85	/	59.06 ± 25.61	32.08
		LAMP [25]	98.85	1.71	85.13 ± 0.31	6.01
		SuRP [20]	98.56	/	86.71	4.43
		Ours	98.85	2.02	$\mathbf{87.7 \pm 0.24}$	**3.44**
ImageNet	VGG-16-BN (Dense: 73.37)	LAMP [25]	95.5	37.16	64.63	8.73
		Ours	95.5	9.12	**66.9**	**6.47**
		Ours	73.54	34.95	**69.35**	**4.02**
		LAMP [25]	98.85	16.73	51.59	21.78
		Ours	89.3	17.71	**68.88**	**4.49**
		Ours	98.85	3.51	**59.41**	**13.96**
	ResNet-50 (Dense: 76.14)	PGMPF [4]	/	53.5	75.11	0.52
		Ours	41	53.5	**75.90**	**0.24**
		LAMP [25]	89.3	26.1	72.56	3.58
		Ours	67.22	28.52	**73.47**	**2.67**
		LAMP [25]	95.5	15.47	66.04	10.1
		Ours	95.5	2.85	**66.06**	**10.08**
		Ours	79.01	16.58	**72.26**	**3.88**
		LAMP [25]	98.85	6.15	42.54	33.61
		Ours	91.41	6.07	**67.91**	**8.23**

Table 1: Iterative Pruning results. **Bold** denotes the highest Top-1 accuracy or lowest accuracy drop among the results with about the same remaining FLOPs; **Red** denotes the highest Top-1 among that with about the same sparsity. Dense denotes the Top-1 accuracy of the unpruned model.



(a) ResNet-32 on CIFAR-10.

(b) DenseNet-121 on CIFAR-10.

(c) VGG-16 on ImageNet.

(d) ResNet-50 on ImageNet.

Figure 2: Iterative pruning process of various classification models and datasets.

architectures and image classification datasets. We consider 3 models on CIFAR-10 dataset [21], *i.e.*, VGG-16 following the adapted architectures in [25], ResNet-32 [17], DenseNet-121 [19], while on ImageNet dataset [8], we evaluate VGG-16 with BatchNorm [34] and ResNet-50 [17]. On CIFAR-10, following the baseline method [25], we perform *five independent trials* for each method, and we report the averages and standard deviations among the trials. On the much larger scaled ImageNet, we only perform one trial for each method. For other implementation details, please refer to the supplementary material.

Details when generating rate-distortion curves. In the experimentations, we need to generate rate-distortion curves for every layer to enable sparsity optimization, where points on the curves are a pair of sparsity level and the model output distortion when certain layer is pruned to that sparsity. For non-data-free scheme, the curves are sampled on a randomly selected calibration set from training dataset, while it is also possible to make it data-free by leveraging synthesized data sampled from certain distribution, *e.g.* standard normal distribution. The size of calibration set is set to 1024 samples for CIFAR-10 and 256 for ImageNet respectively. However, rate-distortion curves obtained by the above process may be interfered by real-world factors resulting in noisy curves. Therefore we designed various strategies to refine the raw rate-distortion curves and better aid the optimization thereafter. Specifically, (1) **Worst case sampling**: inspired by LAMP [25], we calculate the distortion as the *maximum* squared norm error among all calibration samples instead of calculating the *MSE* for the whole calibration set; (2) **Outliers filtering**: treat the local maxima points on the curves that break monotonicity as outlier noises and remove them in order to facilitate Algorithm 1, especially to effectively perform Eq. (12). We provide ablation studies in the later Sec. 4.4 to discuss the individual effects of these strategies.

4.1. Iterative Pruning Results

In the iterative pruning scheme, one starts with a pretrained full-capacity model. During the finetuning process of the pretrained model, we gradually prune out parameters from the model by a certain amount at each iterative stage.

Under different stages of iterative pruning, we get a set of sparse models with gradually increasing sparsity and decreasing computation complexity (FLOPs). Following the iterative pruning settings in LAMP [25], we prune out 20% of the remaining parameters from the model each time after a round of finetuning. The hyper-parameters setup of the finetuning is detailed in the supplementary material. Tab. 1 compares the results of model accuracies produced during the iterative pruning process by our method and other pruning method counterpart.

Given non-standardized adoption of CNN models for experimentations in post-training pruning works, we examined as most models as appeared in various literature and add those as baselines in our comparison, including Global [33], Uniform [45], Uniform+ [13], LAMP [25], E-R ker. [10], which is an extended Erdős-Rényi method for CNNs pruning, where layer-wise sparsity is selected by a closed-form criterion dependent on merely the layer architecture (e.g., the numbers of input and output channels, the convolutional kernel sizes). Fig. 2 further demonstrates the detailed iterative pruning procedures of different methods, where the remaining FLOPs (X-axis) gradually decreases in the course of finetuning.

Results on CIFAR. From Tab. 1, we observe that our method consistently produces pruned models with higher test performance and less test accuracy drop for the same computational complexity (FLOPs) compared to other methods. Fig. 2 further verifies that this observation holds throughout the pruning procedure. For example for ResNet-32 on CIFAR-10, our method obtains Top-1 accuracy at 92.56 on average with 11.25% remaining FLOPs, while baseline result [25] is only 90.04 at 11% FLOPs; When remaining FLOPs is around 3%, we even improve the accuracy by 7.17, *i.e.* only 3.16 accuracy drop with only 4.4% survived parameters. For VGG-16 on CIFAR-10, we also observe similar results, where our method achieves the least accuracy drop among various counterparts, *e.g.*, when FLOPs are within the range of $33 \pm 2\%$, without the advance design of soft gradient masking and weight updating strategies adopted in [4], ours achieves -1.05% drop of Top-1 at 35.49% FLOPs, which means that the pruned network performs better by 1.05% than the unpruned one. PGMPF [4] achieves a higher accuracy score than us on VGG-16 model with 33% remaining FLOPs, which was obtained from a higher performance unpruned model, but still underperforms us regarding the accuracy drop (Top-1 dropped by 0.08%).

Results on ImageNet. On the larger scale dataset ImageNet, we also observe similar behaviors from our approach. For VGG16-BN, we outperform others on both $35\pm2\%$ and 16 ± 2 FLOPs groups. Noticeably, when model sparsity is as high as 98.85%, *i.e.* only 1.15% surviving parameters, our method still has 59.41% accuracy, while LAMP already drops to around 52. This is also observed on ResNet-50, where we outperform LAMP by a large margin at 6% FLOPs group. From Fig. 2c, there is a minor observation that although consistently higher test accuracy with $< 50\%$ FLOPs, VGG-16-BN performs slightly lower within the 30 50% FLOPs range before going up again in the following finetuning iterations. It is speculated that VGG-16-BN is more sensitive to large structural changes for post-train pruning.

In all, for both datasets, we observe that our method generates higher accuracy sparse models given either the same FLOPs or sparsity constraint.

4.2. One-shot Pruning Results

Method	Sparsity (%)	Remaining FLOPs (%)	Top-1 (%)	Top-1 drop(%)
Unpruned	0	100	76.14	-
LAMP [25]	64.5	55	75.43	0.71
OTO [6]	64.5	34.5	75.1	1.04
Ours	58	34.5	**75.59**	**0.55**

Table 2: One-shot pruning results of ResNet-50 on ImageNet.

In one-shot pruning scheme, we directly prune the model to the target computation or parameter constraint, followed by a one-time finetuning. Tab. 2 summarizes the one-shot pruning results using various unstructured pruning algorithms. We carry out comparison on ResNet-50 on ImageNet. The result verifies that our method still fits in the one-shot pruning scheme, with higher accuracy at 34.5% FLOPs than both baselines [25, 6].

4.3. Zero-data Pruning

Method	Sparsity (%)	Remaining FLOPs (%)	Top-1 (%)	Top-1 drop(%)
Unpruned	0	100	76.14	-
[23]	50	/	73.89	2.16
LAMP [25]	50	67.05	74.9	1.24
Ours*	50	42.48	**75.13**	**1.01**

Table 3: Zero-data one-shot pruning results of ResNet-50 on ImageNet dataset. **Ours*** denotes the zero-data alternative of our method by using white noise data to generate rate-distortion curves.

To evaluate whether our method is compatible with zero-data pruning scheme, which is promising to achieve better generalizability than standard pruning schemes that are

usually data dependant, we attempt to adopt our method to zero-data pruning, by replacing the calibration images set that is sampled from real test set with white noise (pixels in each color channel are independently generated by the same distribution required by the classification model, *e.g.*, standard normalized distribution $\mathcal{N}(0, 1)$).

Tab. 3 summarizes the results of zero-data variant of our approach compared to the baseline [23] result with the same data synthesize strategy (white-noise). We also include LAMP [25] in the comparison since it happens to require no calibration set to obtain layerwise pruning thresholds for good. Our approach still achieves superior results under zero-data scenario, with only 1.01% performance drop. This is within our expectation since our rate-distortion theory-based algorithm does not depend on any specific input data distribution.

4.4. Ablation Studies

Arch	Method	Sparsity (%)	Top-1 (%)	Top-1 drop(%)
ResNet-50	Ours	58	**75.59**	**0.55**
	Ours*	60	74.89	1.45
VGG-16-BN	Ours	60	**69.01**	**4.36**
	Ours*	59	62.50	10.87

Table 4: Comparison of joint-optimization objective and vanilla (single-layer) optimization (denoted by Ours*).

WCS	OF	Sparsity (%)	Top-1 (%)	Top-1 drop(%)
Unpruned		0	93.99	-
		89.3	91.15	2.84
√		89.2	91.31	2.68
	√	89	91.51	2.58
√	√	89.3	**92.3**	**1.69**

Table 5: Different post-processing strategies of RD curves on ResNet-32 on CIFAR-10 with iterative pruning scheme. **WCS**: Worst case sampling, **OF**: Outlier filtering.

Since our major contribution is the joint-optimization strategy, we first conducted a comparison with the case not using joint-optimization where we directly solve layer-wise sparsity on the output features of each layer, resulting in the performance shown in Tab. 4. As indicated in the table, we observe deteriorated performances for such single-layer optimization on both tested models, showing that our joint-optimization strategy is optimal.

We also evaluate the individual effectiveness of the aforementioned rate-distortion curves refining strategies.

WCS	OF	Sparsity (%)	Top-1 (%)	Top-1 drop(%)
Unpruned		0	76.14	-
		60	31.22	44.92
√		60	31.22	44.92
	√	60	38.12	38.02
√	√	60	38.12	38.02

Table 6: Different post-processing strategies of RD curves on ResNet-50 on ImageNet with one-shot pruning scheme. Test accuracy of the model **before** finetuning is reported.

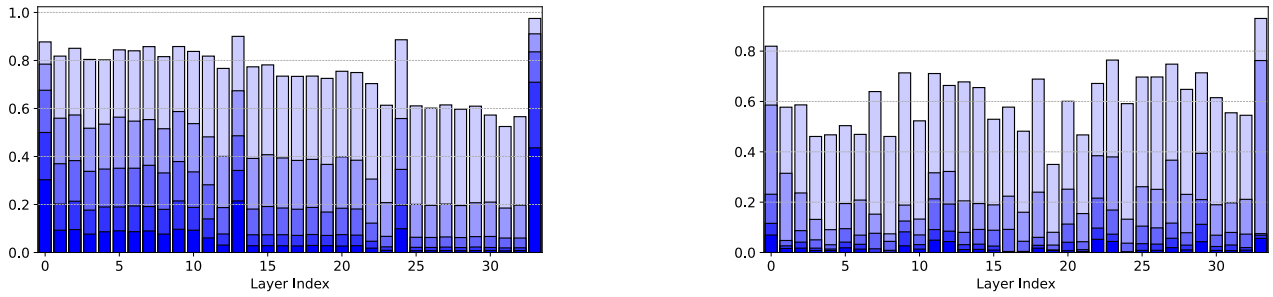
We first perform ablation on CIFAR-10 dataset. From Tab. 5, we observe that at the same model sparsity 89%, which is relatively high for one-shot pruning scheme, both strategies are shown to work positively for our approach. Therefore, we included both strategies to conduct experiments of main results. We also observe the same on ImageNet dataset, as shown in Tab. 6. Particularly, Outlier filtering strategy brings slightly more improvement on both CIFAR-10 and ImageNet, where Worst case sampling makes no difference at this particular sparsity target.

4.5. Other discussions

There is also an interesting observation from Tab. 1 that with the same model sparsity, our method constantly reduces more FLOPs from the model. To better analyze this phenomenon, we take a closer look into the layerwise sparsity solution given by different approaches. As shown in Fig. 3, our method prunes out more parameters from deeper layers than LAMP [25]. Since activations in deeper layers in CNNs usually have more channels and features than shallow layers, pruning out more parameters from deep layers will reduce more operations, resulting in less remaining FLOPs. From Fig. 3, another observation is that both methods prune out more parameters from the last layer of ResNet-32 which is the fully-connected layer, implying that parameters of the last layer contain large redundancy. Meanwhile, we observe that DenseNet-121 on CIFAR-10 does not display the above phenomenon, where our method reduces the same level of FLOPs compared with LAMP under the same sparsity. We elaborate this in the supplementary material.

4.6. Time Complexity

We provide the empirical optimization time complexity analysis in Tab. 7. In practice, we use ternary search algorithm to search the solution of s_i^j in Eq. (12), which has logarithmic time complexity given the search range. On small datasets like CIFAR-10, with 35 layers, our method takes less than a second to calculate the layerwise sparsity,



(a) LAMP. (b) Ours.

Figure 3: Layer-wise sparsity statistics of ResNet-32 on CIFAR-10 of different methods during iterative pruning. Height of bars denotes the pruning rate with $\{0.36, 0.74, 0.89, 0.96, 0.98\}$ model sparsities.

Configuration	No. layers	Sparsity (%)	Time (s)
ResNet-32@CIFAR-10	35	20	0.46 ± 0.09
ResNet-50@ImageNet	54	50	2.08 ± 0.21

Table 7: Time spent on layerwise sparsity optimization of our method.

while on larger ImageNet, our method still only takes a few seconds.

Configuration	Curve Generation (s)	Optimize (s)
ResNet-18@CIFAR-10	1052.64	0.84
VGG-16@CIFAR-10	664.19	2.20

Table 8: Comparison of time cost of RD curve generations and optimization.

We also analyze the curve generation costs. For each layer, we traverse all calibration set samples to calculate output distortion at all sparsity levels to generate a rate-distoriton curve. Therefore, the cost of generating rate-distortion curves becomes $O(lSN)$, where l is the number of layers, S is the number of sparsity levels (we set $S = 100$ in practice), and N is the size of calibration set. We provide the actual time costs for two CIFAR-10 models in Tab. 8. In practice, we used optimized dataloader and parallelized curve generation of different layers to cut down the inference time per sample.

4.7. Analysis of Approximation Error

Given the locality nature of taylor expansion, we expect an increasing discrepancy of the taylor approximation under large distortion. We analyze the empirical approximation error in Fig. 4. The left figure visualizes the relations between the taylor-based approximated output distortion (X-axis) and the real output distortion (Y-axis), we notice that the data points in the figure are very close to the diagonal. The right figure plots the approximation error at different sparsity levels. The approximation error inflates at large sparsities, e.g. $> 50\%$.

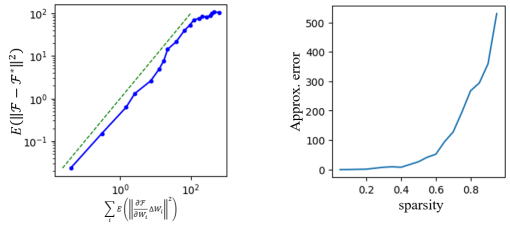


Figure 4: Empirical Approximation Error Analysis.

5. Conclusions

We have presented a new rate-distortion based unstructured pruning criterion. We revealed the output distortion additivity of CNN models unstructured pruning, supported by theory and experiments. We exploited this property to simplify the NP-hard layerwise sparsity optimization problem into a fast pruning criterion with only $O(l \times T^2)$ complexity. Benefiting from the direct optimization on the output distortion, our proposed criterion shows superiority over existing methods in various post-training pruning schemes. Our criterion prefer to prune deep and large layers, leading to significant model size and FLOPs reductions.

Acknowledgement

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its Funds (Project Number A1892b0026 and C211118009 and MTC Programmatic Funds (Grant No. M23L7b0021)). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the A*STAR.

References

[1] Alireza Aghasi, Afshin Abdi, Nam Nguyen, and Justin Romberg. Net-trim: Convex pruning of deep neural networks with performance guarantee. *Advances in neural information processing systems*, 30, 2017.

[2] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018.

[4] Linhang Cai, Zhulin An, Chuanguang Yang, Yangchun Yan, and Yongjun Xu. Prior gradient mask guided pruning-aware fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 1, 2022.

[5] Miguel A Carreira-Perpinán and Yerlan Idelbayev. "learning-compression" algorithms for neural net pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8532–8541, 2018.

[6] Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. *Advances in Neural Information Processing Systems*, 34:19637–19651, 2021.

[7] Pau de Jorge, Amartya Sanyal, Harkirat Behl, Philip Torr, Grégory Rogez, and Puneet K. Dokania. Progressive skeletonization: Trimming more fat from a network at initialization. In *International Conference on Learning Representations*, 2021.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in Neural Information Processing Systems*, 30, 2017.

[10] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[11] Alexander Finkelstein, Uri Almog, and Mark Grobman. Fighting quantization bias with bias. *arXiv preprint arXiv:1906.03193*, 2019.

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

[13] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[14] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

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

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[20] Berivan Isik, Tsachy Weissman, and Albert No. An information-theoretic justification for model pruning. In *International Conference on Artificial Intelligence and Statistics*, pages 3821–3846. PMLR, 2022.

[21] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[22] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[23] Ivan Lazarevich, Alexander Kozlov, and Nikita Malinin. Post-training deep neural network pruning via layer-wise calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 798–805, 2021.

[24] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[25] Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. Layer-adaptive sparsity for the magnitude-based pruning. In *International Conference on Learning Representations*, 2020.

[26] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2018.

[27] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[28] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

[29] Tao Lin, Sebastian U. Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations*, 2020.

[30] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

[31] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[32] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[33] Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *Advances in neural information processing systems*, 32, 2019.

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, May 2015.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[36] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2019.

[37] Huan Wang, Can Qin, Yue Bai, Yulun Zhang, and Yun Fu. Recent advances on neural network pruning at initialization. *arXiv e-prints*, pages arXiv–2103, 2021.

[38] Zhe Wang, Jie Lin, Xue Geng, Mohamed M Sabry Aly, and Vijay Chandrasekhar. Rdo-q: Extremely fine-grained channel-wise quantization via rate-distortion optimization. In *European Conference on Computer Vision*, pages 157–172. Springer, 2022.

[39] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5687–5695, 2017.

[40] Yuxin Zhang, Mingbao Lin, Mengzhao Chen, Fei Chao, and Rongrong Ji. Optg: Optimizing gradient-driven criteria in network sparsity. *arXiv preprint arXiv:2201.12826*, 2022.

[41] Wang Zhe, Jie Lin, Mohamed Sabry Aly, Sean Young, Vijay Chandrasekhar, and Bernd Girod. Rate-distortion optimized coding for efficient cnn compression. In *2021 Data Compression Conference (DCC)*, pages 253–262. IEEE, 2021.

[42] Wang Zhe, Jie Lin, Vijay Chandrasekhar, and Bernd Girod. Optimizing the bit allocation for compression of weights and activations of deep neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3826–3830. IEEE, 2019.

[43] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3599–3608, 2021.

[44] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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