Fast and Controllable Post-training Sparsity: Learning Optimal Sparsity Allocation with Global Constraint in Minutes

Ruihao Gong[1,2], Yang Yong[2], Zining Wang[1], Jinyang Guo[3,1], Xiuying Wei[2], Yuqing Ma[3,1], Xianglong Liu[1*]

[1]State Key Laboratory of Complex & Critical Software Environment, Beihang University
[2]SenseTime Research
[3]Institute of Artificial Intelligence, Beihang University
{gongruihao, 19373122, jinyangguo, mayuqing, xlliu}@buaa.edu.cn, {yongyang, weixiuying}@sensetime.com

Abstract

Neural network sparsity has attracted many research interests due to its similarity to biological schemes and high energy efficiency. However, existing methods depend on long-time training or fine-tuning, which prevents large-scale applications. Recently, some works focusing on post-training sparsity (PTS) have emerged. They get rid of the high training cost but usually suffer from distinct accuracy degradation due to neglect of the reasonable sparsity rate at each layer. Previous methods for finding sparsity rates mainly focus on the training-aware scenario, which usually fails to converge stably under the PTS setting with limited data and much less training cost. In this paper, we propose a fast and controllable post-training sparsity (FCPTS) framework. By incorporating a differentiable bridge function and a controllable optimization objective, our method allows for rapid and accurate sparsity allocation learning in minutes, with the added assurance of convergence to a predetermined global sparsity rate. Equipped with these techniques, we can surpass the state-of-the-art methods by a large margin, e.g., over 30% improvement for ResNet-50 on ImageNet under the sparsity rate of 80%. Our plug-and-play code and supplementary materials are open-sourced at https://github.com/ModelTC/FCPTS.

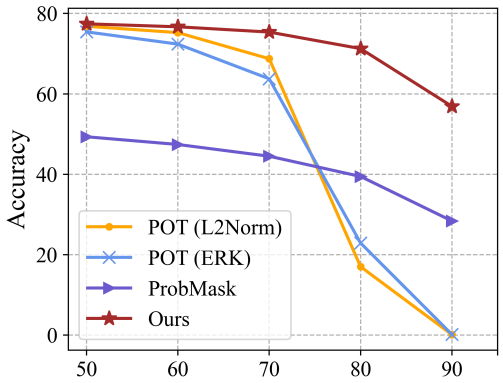

Figure 1: Comparison with existing post-training sparsity methods on ImageNet ResNet-50. Our FCPTS enjoys significant accuracy improvement, especially for the extremely high sparsity rates (e.g., 30% boost under 80% sparsity).

Introduction

Deep neural networks (DNNs) have achieved remarkable success in a variety of fields, including computer vision, natural language processing, and information retrieval. However, when deploying DNNs on resource-limited edge devices, reducing the memory footprint of neural networks and improving energy efficiency become crucial problems. Therefore, various compression techniques are proposed to make the models efficient (Tan et al. 2019; Li et al. 2021b; Wei et al. 2022b; Hinton, Vinyals, and Dean 2015; Jacob et al. 2018; Gong et al. 2019; Li et al. 2021a; Zhu et al. 2020). Among these compression techniques, model sparsity, which prunes the unimportant weights to zero, relates most with the biological brains. Several studies (Hoefler et al. 2021; Jacob et al. 2018; Gopalakrishnan et al. 2018; Liu et al. 2020, 2021; Yuan et al. 2021; Hu et al. 2023) demonstrate that sparsity can contribute to a more robust and generalized model. Additionally, sparsified weights can be stored in special formats that consume less memory. Due to the advantage of low inference cost, low memory requirement, and high generalization ability, model sparsity has attracted much research interest in the community.

However, despite the benefit of inference performance for model sparsification, recovering the accuracy of sparse models is non-trivial. Most methods need to retrain the model for a long time with a large amount of data. It usually takes several hours and even several days when the training dataset is huge, bringing obstacles for large-scale applications in all walks of life. Recently, POT (Lazarevich, Kozlov, and Malinin 2021) proposes to sparsify a neural network in a post-training way without training labels, significantly decreasing the cost of producing a sparse model. They reconstruct the sparse weight layer by layer to improve the accuracy. However, to achieve comparable performance with the dense counterpart, its maximal overall sparsity rate can only reach 50%. When the sparsity level increases, accuracy will crash quickly. This reveals that pushing the limit of post-training sparsity to a higher sparsity level is still challenging. To overcome this challenge, we first identified that the bottleneck of current PTS methods lies in sparsity rate al-

*Corresponding author.

location. The naive non-uniform sparsity allocation does not sufficiently utilize the sparsity sensitivity of each layer. Thus the network performance will degrade greatly as long as the sparse rate exceeds the highest tolerable level that the most sensitive layer can accept.

To solve this problem, many traditional retraining-based approaches are proposed to generate a sparsity rate for each layer, which can be categorized into (1) empirical methods like ERK (Evci et al. 2020), LAMP (Lee et al. 2021), etc. (2) learning-based methods like STR (Kusupati et al. 2020), LTP (Azarian et al. 2020), etc. Both types of methods encounter problems under the PTS setting. The empirical methods heavily depend on handicraft-designed prior knowledge and cannot promise an optimal solution. The learning-based methods either destroy the original weight distribution or require end-to-end training for convergence. Thereby, they can not realize the efficient and accurate sparsity allocation for the post-training case. What's more, many of them need to carefully adjust the hyperparameter and repeat multiple experiments to reach a target sparsity rate, further improving the costs of producing sparse models.

So, the problem remains: *How to design an efficient and effective approach for post-training sparsity?* To address this issue, we propose the Fast and Controllable Post-training Sparsity (FCPTS) framework, which can learn the optimal sparsity allocation with a global constraint in minutes. Specifically, we set up a bridge between the pruning threshold and sparsity rate from the probability density perspective. It is non-trivial to build such a bridge because the sparsity rate calculation is non-differentiable, hindering the backpropagation for sparsity allocation learning. To this end, we use the Kernel Density Estimation (KDE) technique to build this bridge, which can be represented as a differentiable format, making the backpropagation possible. In this way, the sparsity rate of layers across the whole net can be directly optimized in a net-wise pipeline, which is more efficient than the layer-wise pipeline in the work (Lazarevich, Kozlov, and Malinin 2021).

Benefiting from the controllable net-wise optimization, we can obtain the desired sparse neural network in one pass. The whole reconstruction process can be completed within 30 minutes for typical DNNs (e.g., ResNet-18). What's more, equipped with the differentiable modeling of sparsity rate, we can learn the optimal sparsity allocation for each layer without harm to the original weight and thus further unleash the potential for higher sparsity level. With FCPTS, we can produce models with a 70% global sparsity rate with accuracy on par with the dense counterparts. The total process just takes dozens of minutes.

Our FCPTS framework (Figure 2) owns the following advantages compared to state-of-the-art solutions:

1. High efficiency. With FCPTS, the reconstructed sparse network can converge quickly with limited data by net-wise optimization. Thanks to the post-training paradigm and the net-wise optimization pipeline, our FCPTS is extremely efficient. To the best of our knowledge, the time for producing accurate sparse ImageNet models is for the first time reduced to minutes, compared to hours for the existing PTS method.
2. Controllable sparsity allocation. With the help of differentiable FCPTS, the final optimized sparse neural network can promise a specified sparsity rate, releasing the efforts of complex hyperparameter tuning.
3. Accurate and optimal. With the optimized sparsity allocation, we can fully unleash the sparsity potential of each layer in the network and thus generate more accurate sparse models under the same global sparsity rate.
4. Simple and general. FCPTS is easy to implement as a plugin and generalizes on various neural network architectures (ResNet, MobileNet, RegNet, ViT) and tasks (CIFAR-10/100, ImageNet, and PASCAL VOC).

Related Works

Model sparsity dates back decades of years to (1995) which proved that pruning weights based on magnitude was a simple and powerful approach. Nowadays, there are two types of sparsity: structural and non-structural sparsity. In this paper, we focus on the latter one which has a more general format. As for the non-structural sparse models, current methods are mostly based on retraining. Only a few methods sparsify a trained model in the post-training scheme.

Retraining-based and Post-training Sparsity. Various retraining-based methods are proposed in the literature (Ström 1997; Han et al. 2015; Molchanov, Ashukha, and Vetrov 2017; Frankle and Carbin 2019; Renda, Frankle, and Carbin 2020; Narang et al. 2017; Guo et al. 2020; Guo, Xu, and Ouyang 2023; Huang et al. 2023) to improve the accuracy. Although some of them may show promise, they often require extensive training or tuning of hyper-parameters, leading to inefficiency and slow convergence. Recently, a post-training sparsity method (Lazarevich, Kozlov, and Malinin 2021) was proposed, achieving around 50% sparsity rate without significant accuracy degradation. However, its accuracy decreases sharply at higher sparsity levels.

Uniform and Non-uniform Sparsity. The sparsity allocation methods can be categorized as uniform sparsity and non-uniform sparsity. Uniform sparsity methods (Mostafa and Wang 2019; Lin et al. 2020; Gale, Elsen, and Hooker 2019) allocate the same sparsity rate for all the layers, while non-uniform ones assign lower sparsity rates to sensitive layers. Although empirical methods (Evci et al. 2020; Frankle and Carbin 2019; Renda, Frankle, and Carbin 2020) have been used to achieve non-uniform sparsity, they cannot guarantee optimal results as the pruning threshold and sparsity allocation are not jointly learned. Thus there are also methods that aim to learn sparsity by optimizing the mask or pruning threshold, but they all require a long training time and do not work well in post-training settings.

Fast and Controllable Post-training Sparsity Framework

In this section, we introduce our Fast and Controllable Post-training Sparsity (FCPTS) framework. We first define the controllable sparsity reconstruction objective, and then design a differentiable sparsity allocation method without disturbing the trained weights. Based on these, the optimal sparsity allocation can be easily learned.

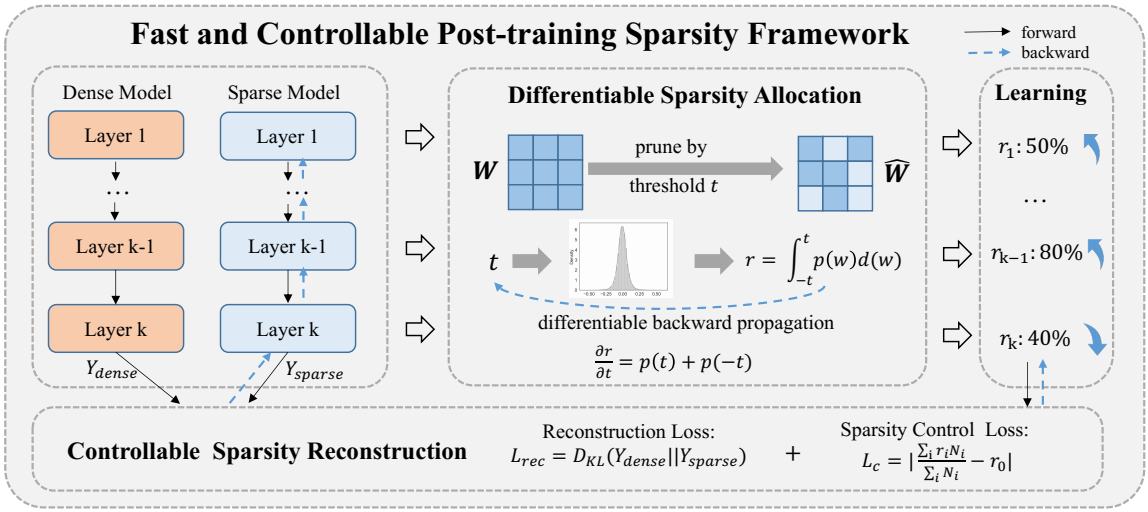

Figure 2: An overview of our fast and controllable post-training sparsity (FCPTS) framework. The differentiable sparsity allocation transfers the learning of the sparsity rate to the threshold by a differentiable estimation, and the controllable sparsity reconstruction enables an optimized network with a specified global sparsity rate. Both components contribute to the final excellent performance.

Preliminary

Mask Generation. Usually, magnitude-based methods have been proven to be simple and effective metrics for mask selection, which can be formulated as follows:

$$\boldsymbol{M} = 0.5 * \text{sgn}(|\boldsymbol{W}| - t) + 0.5, \tag{1}$$

where sgn is the sign function which returns $+1$ for positive values and otherwise returns -1. t is a threshold for pruning weights. Then the sparse model's output can be derived by:

$$\boldsymbol{Y}_{sparse} = f(\boldsymbol{X}, \boldsymbol{M} \odot \boldsymbol{W}). \tag{2}$$

where f denotes the neural network. The mask generation method in Equation 1 implies the fact that weights with larger magnitudes have higher importance for accuracy. This assertion indeed holds for weights in an individual layer.

However, for weights across layers in the whole network, we can not compare them directly due to the huge difference in weight norm. In other words, different layers also have different contributions to the final accuracy. To this end, finding the most suitable sparsity rate for each layer becomes important, especially for the post-training scenario.

Controllable Sparsity Reconstruction

In this part, we present our formulation for controllable sparsity reconstruction, which defines the optimization objective of our FCPTS framework. It incorporates a control loss to achieve a target sparsity rate and employs reconstruction supervision to minimize the difference between the sparse output and dense counterpart.

First, we will introduce the control loss. As mentioned earlier, existing methods mainly use regularization to adjust the weight magnitude, then indirectly influence the sparsity rate. This results in a complicated process to achieve the desired sparsity rate and prevents us from efficiently obtaining a sparse neural network in a limited time. To address this problem, we construct a control loss as defined as:

$$L_c = |\frac{\sum_i r_i N_i}{\sum_i N_i} - r_0|, \tag{3}$$

where the r_i denotes the sparsity rate and N_i is element number of weights of the i_{th} layer. r_0 is the global sparsity rate target and L_c is the loss for controlling the global sparsity rate. With this objective L_c, we can reach the target sparsity rate r_0 without complex hyperparameter tuning.

Furthermore, accompanied by the control loss, we employ a weight reconstruction technique, commonly used in post-training quantization (Wei et al. 2022a), to reduce the difference between sparse and dense output.

$$L_{rec} = D_{\text{KL}}(\boldsymbol{Y}_{dense} \| \boldsymbol{Y}_{sparse}), \tag{4}$$

$D_{\text{KL}}(\cdot)$ represents the Kullback–Leibler divergence function. Under the supervision of the reconstruction loss L_{rec}, the weight optimization will be guided in a direction that contributes to a sparse output closely resembling the dense output and naturally a higher accuracy. What's more, the overall reconstruction is based on the well-trained dense weights, and thus the process is fast and easy to converge.

Finally, we combine L_c and L_{rec} to determine the appropriate sparsity rate for each layer while also making minor adjustments to the weights to accommodate the sparsity. The overall optimization objective can be defined as:

$$L = L_{rec} + L_c, \tag{5}$$

Guided by the whole objective, we can find the reconstructed weight $\boldsymbol{W}$ and sparsity rate r_i of each layer that best suits the target sparsity rate r_0.

$$\arg\min_{\boldsymbol{W}, r_i} L \tag{6}$$

Differentiable Sparsity Allocation

After the above optimization objective is defined, we need to make the optimization process possible. To realize this, an ingenious bridge function is designed to calculate the sparsity rate r from the magnitude threshold t. The derivative of this bridge function can be calculated by kernel density estimation. Then the differentiable optimization of sparsity allocation can be achieved by transferring the optimization to t. With the optimal t of each layer being learned, the sparsity rate r of each layer can be obtained.

Given a specific layer l, the derivative of L with respect to r_l is needed to fulfill the differentiable sparsity learning.

$$\frac{\partial L}{\partial r_l} = \frac{\partial L_{rec}}{\partial r_l} + \frac{\partial L_c}{\partial r_l}. \tag{7}$$

For the latter part of Equation 7, it can be easily obtained according to Equation 3.

$$\frac{\partial L_c}{\partial r_l} = \begin{cases} \frac{N_l}{\sum_i N_i}, & \text{if} \frac{\sum_i r_i N_i}{\sum_i N_i} > r_0, \\ -\frac{N_l}{\sum_i N_i}, & \text{otherwise.} \end{cases} \tag{8}$$

Then we mainly focus on the former part.

$$\frac{\partial L_{rec}}{\partial r_l} = \frac{\partial L_{rec}}{\partial \boldsymbol{M}_l} \cdot \frac{\partial \boldsymbol{M}_l}{\partial t_l} \cdot \frac{\partial t_l}{\partial r_l} \tag{9}$$

Combined with Equation 1, Equation 2, and Equation 4, the first two items $\frac{\partial L_{rec}}{\partial \boldsymbol{M}_l}$ and $\frac{\partial \boldsymbol{M}_l}{\partial t_l}$ are easy to calculate. Thereby the core problem is to calculate $\frac{\partial t_l}{\partial r_l}$. If we can find a differentiable function g that lets $t_l = g(r_l)$. Then the computation of Equation 9 will be feasible.

Bridge function g^{-1} from t_l to r_l: Unfortunately, a differentiable g is hard to construct. But the inverse function g^{-1} can be formalized in a format with differentiable estimation. Then we can transfer the learning of sparsity rate r to threshold t. Given a weight distribution $\boldsymbol{W}_l$ of layer l and its probability density function (PDF) p, w is sampled from the distribution and then the sparsity rate r_l can be got by:

$$r_l = \int_{-t_l}^{t_l} p(w)d(w) = g^{-1}(t_l). \tag{10}$$

Thus the derivative of t_l with respect to r_l can be written as:

$$\frac{\partial r_l}{\partial t_l} = p(t_l) + p(-t_l). \tag{11}$$

Differentiable Estimation: From Equation 11, the key step to calculate the derivative is modeling the probability distribution p. To this end, we utilize the kernel density estimation (KDE) method to estimate the PDF p.

$$p(w) = \frac{1}{n}\sum_i^n K_h(w - w_i) = \frac{1}{nh}\sum_i^n K(\frac{w - w_i}{h}). \tag{12}$$

where K is a non-negative kernel function. n is the number of sampled points. $h > 0$ is a smoothing parameter called the bandwidth. A kernel with subscript h is called the scaled kernel and defined as $K_h(w) = \frac{1}{h}K(\frac{w}{h})$. Intuitively we want to choose h as small as the data will allow.

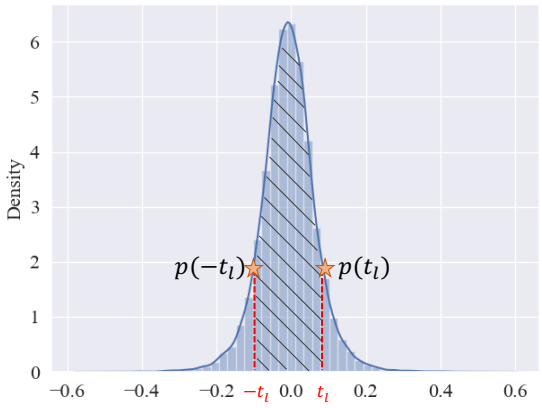

Figure 3: Explanation of bridge function g^{-1} from threshold t_l to sparsity r_l (Equation 10). The area of the shading region equals the sparsity rate r_l. The derivative of r with respect to t can be represented as $p(-t_l) + p(t_l)$.

However, there is always a trade-off between the bias of the estimator and its variance. Here we adopt a commonly used setting: $n = 100, h = 0.5$ and $K(x) = \Phi(x)$, where Φ is the standard normal density function. With this KDE technique, the bridge function is proved to be differentiable.

Transfer the learning of r to t: Equipped with the above bridge function and differentiable estimation, we can optimize the sparsity rate r via the proxy variable t:

$$\frac{\partial L_c}{\partial t_l} = \frac{\partial L_c}{\partial r_l} \cdot \frac{\partial r_l}{\partial t_l}. \tag{13}$$

According to Equation 8, Equation 11 and Equation 12, the derivative of L_c with respect to t_l can be obtained. As for the reconstruction loss, the derivative of L_{rec} with respect to t_l can also be directly calculated:

$$\frac{\partial L_{rec}}{\partial t_l} = \frac{\partial L_{rec}}{\partial \boldsymbol{M}_l} \cdot \frac{\partial \boldsymbol{M}_l}{\partial t_l}. \tag{14}$$

Finally, with the learned threshold t and bridge function Equation 10, we can calculate the exact r, contributing to the flexible and controllable learning of sparsity allocation.

Discussion

Benefiting from the controllable and differentiable sparsity allocation, our FCPTS performs well for the post-training scenario and can reconstruct a sparse neural network with high accuracy as quickly as possible.

Superiority over existing retraining-based non-uniform methods. STR (2020) utilizes $\hat{\boldsymbol{W}} = relu(\boldsymbol{W} - t)$ to learn the threshold t. However, for weights $> t$, the original weight value will be damaged (subtracted by t), hindering the fast accuracy recovery. In contrast, our method will fully utilize the dense weights without damaging their original distribution, contributing to an effective post-training reconstruction. Also STR can not constrain the global sparsity rate into the target value we want. Without careful regularization tuning, the sparsity rate could quickly spiral out of control. ProbMask (Zhou et al. 2021) optimizes the mask with

Method	Sparsity Rate(%)		
	50	60	70
POT	70.1/90.12	68.40/89.10	63.72/86.51
STR	✗	✗	✗
ProbMask	51.97/79.65	49.64/78.19	46.99/76.29

Table 1: Top1/Top5 accuracy of ResNet-18 on ImageNet with different sparsity rates. The traditional learning-based non-uniform sparsity methods fail to work well for the post-training setting, either crashing or converging unstably.

Algorithm 1: FCPTS Framework.

Input: Calibration dataset $\mathcal{D}$, target sparsity rate r_0, a well-trained dense network with weight $\boldsymbol{W}$.
for $d \in \mathcal{D}$ **do**
  {1. Forward propagation:}
  $\boldsymbol{M} = 0.5 * sgn(|\boldsymbol{W}| - t) + 0.5$,
  $\hat{\boldsymbol{W}} = \boldsymbol{M} \odot \boldsymbol{W}$,
  $\boldsymbol{Y}_{dense} = f(\boldsymbol{X}, \boldsymbol{W}), \boldsymbol{Y}_{sparse} = f(\boldsymbol{X}, \hat{\boldsymbol{W}})$,
  Calculate the L_{rec} by Equation 4;
  Obtain the sparsity rate r by t using Equation 10,
  Calculate the L_c by Equation 3;
  Get the overall $L = L_{rec} + L_c$.
  {2. Backward propagation:}
  Calculate $p(w)$ by kernel density estimation in Equation 12,
  Obtain the derivative of r_l with respect to t_l by Equation 11,
  Compute the gradient of t by Equation 8 and Equation 9,
  Adjust the weight and sparsity rate by gradient descent ;
return sparse networks with target sparsity rate r_0 ;

Gumbel Softmax trick. The large optimization space and the stochastic optimization method require extensive steps and huge GPU memory to stably converge. For post-training sparsity, ProbMask is unstable and usually fails to converge to a good solution in a limited time. Table 1 illustrates experimental evidence about the poor performance of existing methods under the PTS setting. The two retraining-based methods STR and ProbMask even underperform the naive post-training method POT (Lazarevich, Kozlov, and Malinin 2021).

Efficiency. FCPTS enjoys high efficiency since it reaches the target sparsity rate with only one pass of net-wise reconstruction, instead of layer-by-layer progressive optimization in POT. Usually, it can generate sparse neural networks in dozens of minutes using just one NVIDIA RTX 3090 GPU. The overall pipeline is summarized in Algorithm 1.

Experiment

In this section, we conduct a series of experiments to evaluate the effect of FCPTS. We first conduct extensive experiments on image classification and object detection to compare with existing methods. Then in-depth analyses are given to reveal the internal superiority of our method.

Results on Various Tasks

We conduct comprehensive validations on four datasets covering image classification (CIFAR-10/100 (Krizhevsky, Hinton et al. 2009), ImageNet (Russakovsky et al. 2015)) and object detection (PASCAL VOC (Everingham et al. 2010)). The CIFAR-10 dataset consists of 50K training images and 10K testing images of size 32×32 with 10 classes. ImageNet ILSVRC12 contains about 1.2 million training images and 50K testing images with 1,000 classes. The PASCAL VOC is a widely used object detection dataset containing 20 object categories. Each image in this dataset has bounding box annotations and object class annotations.

Baseline Setting. We choose the only PTS method POT (Lazarevich, Kozlov, and Malinin 2021) as our baseline. The calibration dataset contains 10k images. Also, we reproduced some techniques originally designed for retraining, i.e., the heuristic non-uniform sparsity method ERK in POT, and ProbMask under the PTS setting.

Network Structures. We employ the widely-used network structures including ResNet-32/56 for CIFAR-10/CIFAR-100, and ResNet-18/50 (He et al. 2016), MobileNetV2 (Sandler et al. 2018), RegNet-200M/400M (Radosavovic et al. 2020), Vit-Base/Large (Dosovitskiy et al. 2021) for ImageNet, and MobileNetV1 SSD, MobileNetV2 SSD-lite (Liu et al. 2016) for PASCAL VOC.

CIFAR-10/CIFAR-100 Table 2 clearly shows that our method outperforms the baselines. As the sparsity increases, the superiority becomes even more significant. The baseline method degrades rapidly at an 80% sparsity and collapses completely at 90% or earlier. In contrast, our FCPTS remains stable at a very high accuracy even at a sparsity of 90%. This is particularly evident in the CIFAR-100 dataset.

ImageNet The accuracy results for various sparsity rates on ImageNet are presented in Table 3. Our method is shown to outperform the baseline with a significant margin across all models, particularly at sparsity rates exceeding 70%. Notably, at a sparsity rate of 70%, our method achieves accuracy levels almost similar to its dense counterpart on ResNet-18 and ResNet-50 (with a mere 2% degradation), while existing methods experience a significant accuracy loss of near 10%. For mobile-friendly architectures like RegNet and MobileNet, existing methods encounter unacceptable accuracy decreases under 60% sparsity, whereas our method can improve by 3%-8%. These results provide detailed evidence of the advantages of our proposed method. Furthermore, we tested the effect on ViT models in Table 4. The consistent improvement proves that our FCPTS also generalizes for attention-based architectures.

PASCAL VOC We report the detailed accuracy from two variants of SSD at the sparsity rate 90% in Table 5. Compared with the POT using L2-normalization magnitude and ERK sparsity rate allocation algorithm, our FCPTS can get a significant improvement under this extreme conditions of 90% sparsity rate. For the sparse MobileNetV1 SSD, the mAP of our FCPTS only drops a little. For the compressed

Model	Method	CIFAR-10			CIFAR-100		
		70	80	90	70	80	90
ResNet-32	POT (L2Norm)	91.06/99.64	84.80/98.85	20.09/71.73	59.35/86.47	28.98/63.09	3.01/11.55
	POT (ERK)	90.95/99.62	84.91/98.59	21.87/74.36	56.21/84.71	29.50/62.64	4.66/14.30
	Ours	**92.91/99.77**	**92.36/99.73**	**90.35/99.65**	**69.14/91.07**	**68.28/90.87**	**63.35/89.38**
ResNet-56	POT (L2Norm)	93.01/99.74	87.18/98.83	28.21/72.24	64.63/88.90	36.19/69.07	4.41/13.00
	POT (ERK)	92.84/99.70	87.03/98.47	37.70/70.33	62.65/88.16	36.96/69.44	5.88/17.37
	Ours	**94.07/99.83**	**93.67/99.78**	**92.02/99.78**	**72.16/92.02**	**71.09/91.90**	**68.07/90.51**

Table 2: Comparison of the Top1/Top5 accuracy (%) on ResNet-32/56 under different sparsity rates on dataset CIFAR-10/100. The accuracies (%) of dense ResNet-32 on dataset CIFAR-10/100 are 93.53/99.77 and 70.16/90.89, respectively. The accuracies (%) of dense ResNet-56 on dataset CIFAR-10/100 are 94.37/99.83 and 72.63/91.94, respectively.

Model	Method	Sparsity Rate (%)				
		50	60	70	80	90
ResNet-18 70.88/90.45	POT (L2Norm)	70.06/89.12	68.40/89.10	63.72/86.51	44.94/71.51	6.03/15.70
	POT (ERK)	69.66/89.40	68.24/88.62	64.28/86.40	50.78/75.59	5.66/15.81
	ProbMask	51.97/79.65	49.64/78.19	46.99/76.29	42.58/72.58	32.88/64.33
	Ours	**70.07/89.61**	**69.58/89.37**	**68.50/88.89**	**65.83/87.69**	**57.40/83.37**
ResNet-50 77.89/93.762	POT (L2Norm)	76.83/93.81	75.24/93.22	68.74/89.44	17.03/30.31	0.13/0.42
	POT (ERK)	75.43/92.51	72.38/90.89	63.76/84.74	22.92/42.13	0.17/1.25
	ProbMask	49.34/78.42	47.42/76.88	44.50/74.95	39.47/70.97	28.40/60.22
	Ours	**77.43/93.55**	**76.69/93.38**	**75.38/92.87**	**71.26/91.23**	**56.91/84.86**
RegNetX-200M 68.41/89.11	POT (L2Norm)	64.69/87.08	59.98/84.40	48.36/75.86	24.48/49.90	2.02/7.42
	POT (ERK)	64.54/86.44	60.71/84.40	52.97/79.46	31.56/60.02	1.36/4.83
	ProbMask	50.23/77.66	47.67/75.97	44.61/73.71	39.43/69.78	29.87/60.26
	Ours	**66.57/87.30**	**65.30/86.73**	**62.80/85.32**	**57.44/82.37**	**40.90/71.47**
RegNetX-400M 71.84/90.55	POT (L2Norm)	67.54/88.62	65.68/87.73	56.63/81.66	28.55/54.41	2.32/7.87
	POT (ERK)	69.97/89.59	67.07/88.25	60.40/84.22	37.10/65.71	1.23/3.53
	ProbMask	48.84/77.17	46.25/75.54	43.20/73.15	38.59/69.14	30.30/60.96
	Ours	**70.85/90.17**	**70.09/89.73**	**68.27/89.13**	**64.23/87.15**	**50.00/78.97**
MobileNetV2 72.85/91.61	POT (L2Norm)	65.51/87.31	57.95/81.35	21.60/44.53	0.22/1.47	0.10/0.55
	POT (ERK)	69.80/89.66	64.98/87.14	49.14/76.25	9.75/25.30	0.21/0.77
	ProbMask	30.24/63.68	25.15/57.52	18.92/48.66	11.76/35.48	4.41/16.75
	Ours	**70.52/90.07**	**68.10/89.16**	**61.18/86.04**	**40.20/74.28**	**13.99/39.75**

Table 3: Comparison of the Top1/Top5 accuracy (%) on various CNNs under different sparsity rates on ImageNet dataset. The accuracy (%) of the dense model is listed under the architecture of the model.

Model	Method	Sparsity Rate (%)		
		50	60	70
ViT base 75.68/92.94	POT(L2Norm)	62.08/83.92	53.63/77.08	29.96/53.16
	POT(ERK)	68.04/89.91	61.67/71.48	30.50/55.30
	Ours	**74.90/92.99**	**72.09/91.47**	**65.24/87.74**
ViT large 79.29/94.78	POT(L2Norm)	77.11/93.91	75.33/93.06	70.42/90.78
	POT(ERK)	77.46/94.22	75.64/93.33	71.02/91.23
	Ours	**78.13/94.41**	**76.71/93.73**	**73.01/92.07**

Table 4: Comparison of the Top1/Top5 accuracy (%) on ViT models under different sparsity rates on ImageNet dataset.

Model	Method	mAP (%)
MobileNetV1 SSD 67.7	POT (L2Norm)	48.5
	POT (ERK)	54.6
	Ours	**65.1**
MobileNetV2 SSD-Lite 68.6	POT (L2Norm)	16.4
	POT (ERK)	0.3
	Ours	**59.1**

Table 5: Comparison on PASCAL VOC at 90% sparsity.

MobileNetV2 SSD-Lite model, our FCPTS can get 59.1% mAP while the performances of other methods crash.

Effect of Learnable Sparsity

In this section, we study how the learnable sparsity contributes to the final performance. We just initialize the sparsity allocation of models with ERK and fix the sparsity rates of all layers, then reconstruct with or without learning the sparsity rate. We conducted this experiment on three models, ResNet-50, RegNetX-400M, and MobileNetV2. Figure 4 shows that models with learnable sparsity can get higher accuracy compared with models without learnable sparsity. This gap in accuracy is more pronounced at high sparsity rates, e.g., about 10% improvement for RegNetX-400M under the sparsity rate of 80%. This experiment strongly demonstrates that the learnable sparsity with a global constraint can dynamically optimize the sparsity allocation to make it more reasonable and rescue it from some unexpected sparsity allocation traps.

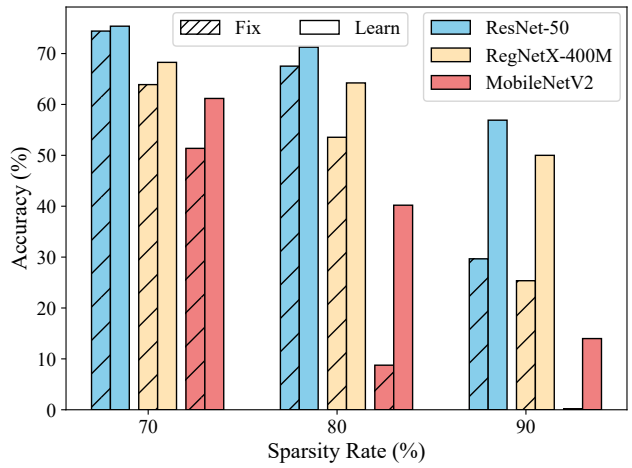

Figure 4: The effect of learnable sparsity. For each neural network, the left dash bars are results with fixed sparsity rate and the right bars are results with learned sparsity rate.

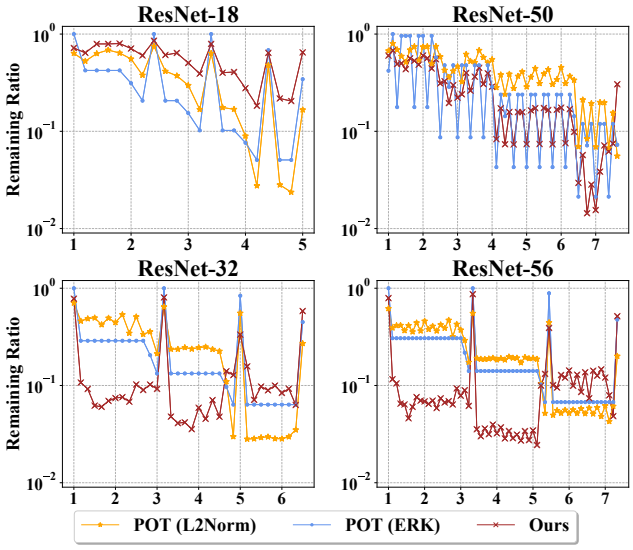

Figure 5: Visualization of the optimized sparsity allocation at a sparsity rate of 90%. ResNet-18 and ResNet-50 are on ImageNet, ResNet-32 and ResNet-56 are on CIFAR-100.

Sparsity Allocation Analyses

In Figure 5, we prove the effectiveness of FCPTS which can lead to more reasonable sparsity allocation. The optimized sparsity allocation for 4 models reconstructed on the ImageNet and CIFAR-100 datasets is presented. It can be found that our method effectively learns to allocate lower sparsity for the latter layers and increase sparsity for some other layers to reach the global sparsity rate target. Additionally, the first layer exhibits a relatively high sparsity rate, in alignment with experience in previous works. Without the learned allocation, the other two methods tend to exceed the sparsity tolerance and suffer a huge accuracy crash.

Method	Dataset	Model	Time (min)
POT	CIFAR-100	ResNet-32	31
	ImageNet	ResNet-18	100
Ours	CIFAR-100	ResNet-32	9
	ImageNet	ResNet-18	29

Table 6: Efficiency for generating a sparse neural network.

Model	Sparsity rate (%)	Latency (ms)	Speed up	Memory (MB)
ResNet-18	dense	18.194	-	11.632
	50	10.796	1.695	7.640
	60	9.287	1.959	6.530
	70	7.599	2.394	5.245
MobileNetV2	dense	33.936	-	6.207
	50	28.387	1.195	6.488
	60	28.157	1.205	5.686
	70	26.863	1.263	5.235

Table 7: Inference performance of sparse ResNet-18 and MobileNetV2 on CV22, a hardware of autonomous driving.

Efficiency

To present the high efficiency of our method, we collect the reconstruction time cost to obtain a sparse neural network and evaluate the inference speed on the real hardware.

Reconstruction Efficiency. Since FCPTS obeys the original weight distribution of the dense model, it can fully take advantage of the historical training efforts. Besides, it directly adopts net-wise optimization on a reduced optimization space (i.e., threshold) without any randomness and without complex layer-wise calculation for MSE. So it can converge quickly and stably. More importantly, the controllable sparsity rate enables us to reach the target without repeated hyper-parameter tunings. From Table 6, we can see that our FCPTS for the first time pushes the time for generating ImageNet sparse models into minutes while the existing state-of-the-art method POT requires over one and a half hours. Our framework enjoys a 3 times speedup.

Inference Efficiency. We also test the inference performance of the models sparsified by FSPTS on Ambarella CV22, an autonomous driving chip supporting the acceleration of unstructured sparsity. As seen in Table 7, benefiting from the sparsity, both inference latency and memory occupation are significantly reduced. This gain is especially remarkable for the ResNet-18, which achieves nearly 2.4x speedup and over 50% memory saving at 70% sparsity.

Conclusion

In this paper, we propose a fast and controllable post-training sparsity (FCPTS) framework that pushes the limit of PTS accuracy to a new level. It utilizes a differentiable estimation to enable a learnable and controllable sparsity rate. Benefiting from the optimal sparsity allocation, our FCPTS achieves state-of-the-art results on 4 different datasets covering classification and object detection tasks.

Acknowledgments

This work was supported in part by the National Key Research and Development Plan of China (2022ZD0116405), the National Natural Science Foundation of China (No. 62206010, No.62022009, No. 62306025), and the State Key Laboratory of Software Development Environment (SKLSDE-2022ZX-23).

References

Azarian, K.; Bhalgat, Y.; Lee, J.; and Blankevoort, T. 2020. Learned threshold pruning. *arXiv preprint arXiv:2003.00075*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Evci, U.; Gale, T.; Menick, J.; Castro, P. S.; and Elsen, E. 2020. Rigging the Lottery: Making All Tickets Winners. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 2943–2952. PMLR.

Everingham, M.; Gool, L. V.; Williams, C. K. I.; Winn, J. M.; and Zisserman, A. 2010. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.*, 88(2): 303–338.

Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.

Gale, T.; Elsen, E.; and Hooker, S. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

Gong, R.; Liu, X.; Jiang, S.; Li, T.; Hu, P.; Lin, J.; Yu, F.; and Yan, J. 2019. Differentiable Soft Quantization: Bridging Full-Precision and Low-Bit Neural Networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

Gopalakrishnan, S.; Marzi, Z.; Madhow, U.; and Pedarsani, R. 2018. Combating adversarial attacks using sparse representations. *arXiv preprint arXiv:1803.03880*.

Guo, J.; Xu, D.; and Ouyang, W. 2023. Multidimensional Pruning and Its Extension: A Unified Framework for Model Compression. *IEEE Transactions on Neural Networks and Learning Systems*.

Guo, J.; Zhang, W.; Ouyang, W.; and Xu, D. 2020. Model compression using progressive channel pruning. *IEEE Transactions on Circuits and Systems for Video Technology*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hoefler, T.; Alistarh, D.; Ben-Nun, T.; Dryden, N.; and Peste, A. 2021. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *arXiv:2102.00554 [cs]*. ArXiv: 2102.00554.

Hu, W.; Che, Z.; Liu, N.; Li, M.; Tang, J.; Zhang, C.; and Wang, J. 2023. : Channel Pruning via Class-Aware Trace Ratio Optimization. *IEEE Transactions on Neural Networks and Learning Systems*.

Huang, Y.; Liu, N.; Che, Z.; Xu, Z.; Shen, C.; Peng, Y.; Zhang, G.; Liu, X.; Feng, F.; and Tang, J. 2023. CP3: Channel Pruning Plug-In for Point-Based Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5302–5312.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2704–2713.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kusupati, A.; Ramanujan, V.; Somani, R.; Wortsman, M.; Jain, P.; Kakade, S.; and Farhadi, A. 2020. Soft Threshold Weight Reparameterization for Learnable Sparsity. In *Proceedings of the International Conference on Machine Learning*.

Lazarevich, I.; Kozlov, A.; and Malinin, N. 2021. Post-training deep neural network pruning via layer-wise calibration. ArXiv:2104.15023 [cs].

Lee, J.; Park, S.; Mo, S.; Ahn, S.; and Shin, J. 2021. Layer-adaptive Sparsity for the Magnitude-based Pruning. In *International Conference on Learning Representations*.

Li, Y.; Gong, R.; Tan, X.; Yang, Y.; Hu, P.; Zhang, Q.; Yu, F.; Wang, W.; and Gu, S. 2021a. {BRECQ}: Pushing the Limit of Post-Training Quantization by Block Reconstruction. In *International Conference on Learning Representations*.

Li, Y.; Shen, M.; Ma, J.; Ren, Y.; Zhao, M.; Zhang, Q.; Gong, R.; Yu, F.; and Yan, J. 2021b. MQBench: Towards Reproducible and Deployable Model Quantization Benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Lin, T.; Stich, S. U.; Barba, L.; Dmitriev, D.; and Jaggi, M. 2020. Dynamic Model Pruning with Feedback. In *International Conference on Learning Representations*.

Liu, N.; Ma, X.; Xu, Z.; Wang, Y.; Tang, J.; and Ye, J. 2020. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 4876–4883.

Liu, N.; Yuan, G.; Che, Z.; Shen, X.; Ma, X.; Jin, Q.; Ren, J.; Tang, J.; Liu, S.; and Wang, Y. 2021. Lottery Ticket Preserves Weight Correlation: Is It Desirable or Not? In *International Conference on Machine Learning*, 7011–7020. PMLR.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. SSD: Single Shot MultiBox Detector. In *ECCV*.

Molchanov, D.; Ashukha, A.; and Vetrov, D. 2017. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, 2498–2507. PMLR.

Mostafa, H.; and Wang, X. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 4646–4655. PMLR.

Narang, S.; Diamos, G.; Sengupta, S.; and Elsen, E. 2017. Exploring Sparsity in Recurrent Neural Networks. In *International Conference on Learning Representations*.

Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; and Dollár, P. 2020. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10428–10436.

Renda, A.; Frankle, J.; and Carbin, M. 2020. Comparing fine-tuning and rewinding in neural network pruning. In *International Conference on Learning Representations*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ström, N. 1997. Sparse connection and pruning in large dynamic artificial neural networks. In *Fifth European Conference on Speech Communication and Technology*. Citeseer.

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

Thimm, G.; and Fiesler, E. 1995. Evaluating pruning methods. In *Proceedings of the International Symposium on Artificial neural networks*, 20–25.

Wei, X.; Gong, R.; Li, Y.; Liu, X.; and Yu, F. 2022a. QDrop: Randomly Dropping Quantization for Extremely Low-bit Post-Training Quantization. In *International Conference on Learning Representations*.

Wei, X.; Zhang, Y.; Zhang, X.; Gong, R.; Zhang, S.; Zhang, Q.; Yu, F.; and Liu, X. 2022b. Outlier Suppression: Pushing the Limit of Low-bit Transformer Language Models. In *Thirty-Sixth Conference on Neural Information Processing Systems*.

Yuan, G.; Ma, X.; Niu, W.; Li, Z.; Kong, Z.; Liu, N.; Gong, Y.; Zhan, Z.; He, C.; Jin, Q.; et al. 2021. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34: 20838–20850.

Zhou, X.; Zhang, W.; Xu, H.; and Zhang, T. 2021. Effective sparsification of neural networks with global sparsity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3599–3608.

Zhu, F.; Gong, R.; Yu, F.; Liu, X.; Wang, Y.; Li, Z.; Yang, X.; and Yan, J. 2020. Towards Unified INT8 Training for Convolutional Neural Network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.