UniPTS: A Unified Framework for Proficient Post-Training Sparsity

Jingjing Xie[1], Yuxin Zhang[1], Mingbao Lin[2], Zhihang Lin[1], Liujuan Cao[1]*, Rongrong Ji[1]
[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing,
Ministry of Education of China, School of Informatics, Xiamen University.
[2]Tencent Youtu Lab
{jingjingxie, yuxinzhang, lmbxmu, zhihanglin}@stu.xmu.edu.cn,
{caoliujuan, rrji}@xmu.edu.cn

Abstract

Post-training Sparsity (PTS) is a recently emerged avenue that chases efficient network sparsity with limited data in need. Existing PTS methods, however, undergo significant performance degradation compared with traditional methods that retrain the sparse networks via the whole dataset, especially at high sparsity ratios. In this paper, we attempt to reconcile this disparity by transposing three cardinal factors that profoundly alter the performance of conventional sparsity into the context of PTS. Our endeavors particularly comprise (1) A base-decayed ***sparsity objective*** *that promotes efficient knowledge transferring from dense network to the sparse counterpart. (2) A reducing-regrowing search algorithm designed to ascertain the optimal* ***sparsity distribution*** *while circumventing overfitting to the small calibration set in PTS. (3) The employment of dynamic sparse training predicated on the preceding aspects, aimed at comprehensively optimizing the* ***sparsity structure*** *while ensuring training stability. Our proposed framework, termed UniPTS, is validated to be much superior to existing PTS methods across extensive benchmarks. As an illustration, it amplifies the performance of POT, a recently proposed recipe, from 3.9% to 68.6% when pruning ResNet-50 at 90% sparsity ratio on ImageNet. We release the code of our paper at https://github.com/xjjxmu/UniPTS.*

1. Introduction

Deep neural networks (DNNs) have shown exceptional performance in a variety of tasks, including computer vision [3, 4, 36], natural language processing [6, 33], *etc.* However, this extraordinary growth is offset by an overwhelming volume of model parameters, ranging from several millions to billions [2, 42], presenting an impediment to DNN implementation in resource-constrained contexts. Accordingly, prodigious efforts have been channeled towards the evolution of model compression algorithms, encompassing model quantization [7, 29], network sparsity [8, 12], and knowledge distillation [15, 41].

*Corresponding Author

Notwithstanding the proficiency of these methods in shrinking the size of DNNs, they typically necessitate a model retraining phase via the full training set to recover the performance. This could prove exceedingly burdensome in situational contexts marked by constraints both in terms of training resources and the accessibility of the dataset. To circumvent this obstacle, researchers have developed post-training compression methodologies, which efficiently tune the compressed DNNs using a comparatively petite calibration dataset. Most notable advancements to date have predominantly revolved around Post-Training Quantization (PTQ), where the quantized 8-bit model can reach performance on par with its full-precision counterpart [10]. However, Post-Training Sparsity (PTS) has garnered relatively little attention, notwithstanding the commensurate prominence of sparsity in comparison with quantization for compressing DNNs.

One possible interpretation implicates that network sparsity methods rely more heavily on weight retraining, *w.r.t.*, iterative fine-tuning [12], or even training from scratch [8], to recover the performance. Thereby, considerable challenges emerge when merely minimal calibration datasets are accessible for such a retraining phase. A recently proposed recipe, POT [18], sets a standard benchmark for PTS, undertaking a layer-wise minimization of the output discrepancy between sparse and dense weights via Mean Square Error (MSE) loss. Despite its proficiency in preserving the performance at moderate pruning rates such as 50%, POT encounters drastic performance loss at high sparsity rates, particularly deteriorating to the random level at 90% sparsity, where traditional sparsity methods still manage to uphold performance. Therefore, the argument over the necessity for data efficiency and maintenance of performance

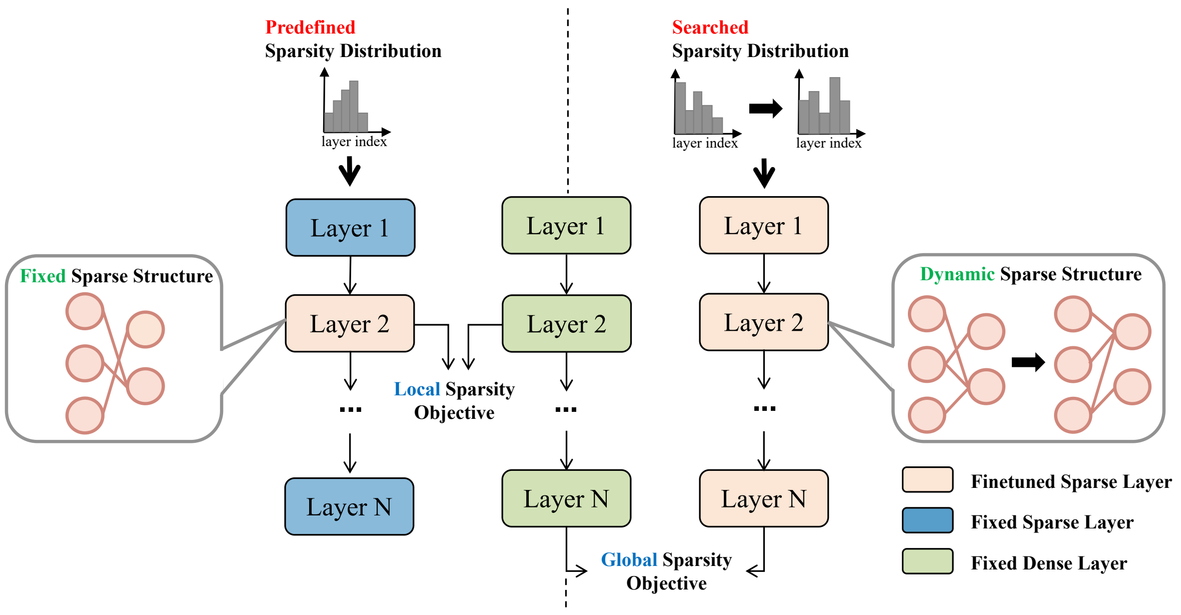

Figure 1. Comparison between POT and our UniPTS framework. **Left** shows that POT uses predefined sparsity distribution to obtain a fixed sparse structure and retrains the pruned layer with the local sparsity objective. But UniPTS(**Right**) searches the optimal sparsity distribution and leverages the global sparsity objective and dynamic sparsity training to explore optimal sparse structures.

within network sparsity persists as an unresolved matter so far.

In this work, we present UniPTS as a practical remedy to ameliorate this issue. As shown in Figure 1, UniPTS is a hybrid approach, meticulously crafted by investigating the failure of conventional sparsity methods in PTS scenarios from three vantage points including the ***sparsity objective***, ***sparsity distribution***, and ***sparsity structure***. These aspects collaboratively contribute to the performance retention of sparse networks [8, 11, 17, 19, 46]. In particular, we first amend the **sparsity objective** from layerwise MSE [21, 33, 40] in POT [18] to a global Kullback-Leibler divergence, with its log base adaptively evolves throughout the training schedule. This not only fosters training acceleration, but also augments supervision from dense networks to sparse counterparts in a fluid manner. Subsequently, we propose a novel evolutionary search algorithm to optimize the ***sparsity distribution***, *i.e.*, layerwise sparsity ratios in PTS. The principle innovations fall into an excessive sparsity allocation mechanism and a noise-disturbed fitness evaluation, which guarantees a robust search for the optimal solution while avoiding over-fitting to the small calibration set. Under the constraints of the preceding sparsity objective and sparsity distribution, we concludingly adopt the concept of Dynamic Sparsity Training (DST) [8, 23, 24] to comprehensively explore the ***sparsity structure*** in PTS instead of retraining the pruned network in a fixed typology [11, 19, 39].

In an expansive series of experiments spanning diverse computer vision tasks, we substantiate the efficacy of our proposed UniPTS. The empirical evidence underscores that our method exhibits enhanced applicability to PTS, materializing a marked augmentation in performance, particularly prominent at elevated sparsity rates. To illustrate, it improves the performance of POT from a meager 3.9% to a considerable 68.6% for pruning ResNet-50 at 90% sparsity ratio, even using less training time. Our work provides fresh insights into boosting the performance of PTS and we hope to encourage more research in probing the advantages of network sparsity through a pragmatic perspective.

2. Related Work

2.1. Post-Training Model Compression

In quantization, various techniques have been developed to reduce the training overhead and data requirement, known as PTQ [10, 29]. However, existing methods often involve time-consuming retraining processes when it comes to sparsity. POT [18] is proposed as a pioneering pipeline for PTS to address this issue. POT suggests an iterative pruning to obtain the sparse network. In order to mitigate bias introduced by pruning, it follows a quantization methodology to correct bias [28] and then fine-tunes the weights via the reconstruction of the feature map layer by layer. Although POT offers an initial insight into PTS, its performance degrades a lot at a high sparsity rate.

2.2. Sparsity Distribution

Given a global sparsity rate, how to obtain an optimal layerwise sparsity is an essential problem. Existing solutions can be categorized into heuristic-based, optimization-based, and search-based methods. Heuristic methods take advantage of the characteristics of each layer. For instance, the Erdos-Renyi-Kernel (ERK) allocates per-layer sparsity based on the number of parameters [8]. Such techniques rely on empirical analysis and cannot guarantee the optimal solution. Optimization-based methods learn the sparsity distribution by optimizing a pruning threshold or learnable masks [17, 30]. Search-based methods aim to discover the optimal sparsity distribution via reinforcement learning [14] or evolutionary algorithms [26]. These methods iteratively explore different sparsity configurations and evaluate their performance. These two kind of methods integrate the task objective with sparsity distribution and provide a flexible framework for sparsity allocation. However, most of them rely on regularization to meet the requirement of the global sparsity and need to tune hyperparameters delicately. We design an efficient evolutionary search for the optimal sparsity distribution. It can meet the requirement about the global sparsity without tuning hyperparameters and alleviate overfitting in PTS.

2.3. Dynamic Sparsity Training

Traditional network sparsity involves two steps: pruning and fine-tuning. Static sparsity training means that the sparse structure of the network remains unchanged after pruning and only the preserved weights are fine-tuned. Representative works of this type of method include one-shot pruning [19] and gradual pruning [12]. Dynamic sparsity training represents that model's weights are dynamically pruned and regrow during fine-tuning. Evci *et al.* [8] suggested the magnitude as the criterion for pruning and the gradient as the criterion for regrowth. In our paper, we find the traditional sparse training is not suitable for PTS, and modify both sparsity objective and training strategy.

3. Method

3.1. Background

Primarily, we elucidate the fundamental concepts of network sparsity. Given the dense weights $\mathbf{W}$, network sparsity can be conceptualized as applying a binary mask $\mathbf{M}$ to $\mathbf{W}$. A zero element within $\mathbf{M}$ signifies the removal of a specific weight, whereas non-zero elements indicate preservation. The zero-masked weights alter network outputs, which consequently results in negligible performance degradation in sparse networks, particularly at high sparsity rates [8, 11].

To preserve the functionality of sparse networks, researchers have explored enhancing sparse networks from various perspectives including: 1) Employing training data to fine-tune the sparse network, with minimizing the discrepancy between the ground-truth labels and network outputs as the *sparsity objective* [12]; 2) Determining the *sparsity distribution* across all layers, *i.e.*, the relative sparsity ratio for each individual layer [17, 26, 30]; 3) Exploring the *sparsity structure* by modeling sparsity training to optimize the binary mask $\mathbf{M}$, therefore determining the removed weights for each layer [23, 24, 46].

Albeit a plethora of methods proposed, they all necessitate substantial volumes of training data and computational costs. In scenarios characterized by limited resources, it becomes imperative to recover the performance of sparse networks solely through restrained quantities of data, referred to as Post-Training Sparsity (PTS) [18]. Regrettably, existing pruning methods demonstrate a deficit in scalability within post-training scenarios, enduring severe performance declines compared with full-data utilization. Actually, the requirement for proficient and high-performing network sparsity reveals an unresolved dilemma in the current landscape.

3.2. UniPTS

In this section, we examine the underlying reasons for the failure of conventional sparsity methods in data-constrained scenarios and subsequently propose innovative UniPTS, a unified framework to ameliorate their shortcomings and achieve proficient PTS. Our efforts are articulated across the previously mentioned three dimensions – *sparsity objective*, *sparsity distribution*, and *sparsity structure* – that are integral to recovering the performance of sparse networks.

3.2.1 Base-Decayed Sparsity Objective

In traditional literature, the sparsity objective simply falls into fine-tuning the sparse network using training data, cost of which is basically the same as pre-training a dense network. As a pioneering pipeline for PTS, POT [18] utilizes the mean-squared error (MSE) loss as the training objective and fine-tunes per-layer weights independently:

$$\mathcal{L}_{\text{MSE}} = ||\mathbf{Y}^l - \hat{\mathbf{Y}}^l||^2, \tag{1}$$

where $\mathbf{Y}^l$ and $\hat{\mathbf{Y}}^l$ denote the l-th layer outputs of dense and sparse layers, respectively. This supervision enables POT to effectively recover performance at moderate sparsity rates; however, its performance at high sparsity levels falls significantly behind traditional fine-tuning with full training data [38, 46].

The above layer-wise MSE is a localized metric that, if not well optimized, gradually gathers bias towards the final task prediction. An increasing sparsity rate challenges the match of individual values of $\mathbf{Y}^l$ and $\hat{\mathbf{Y}}^l$, and gives rise to expanded bias. This analysis correlates with the phenomenon that POT drastically declines to randomized performance when confronted with a 90% sparsity rate.

Given this, we choose to globally optimize the final prediction probability distributions $\mathbf{Z}$ from the dense output

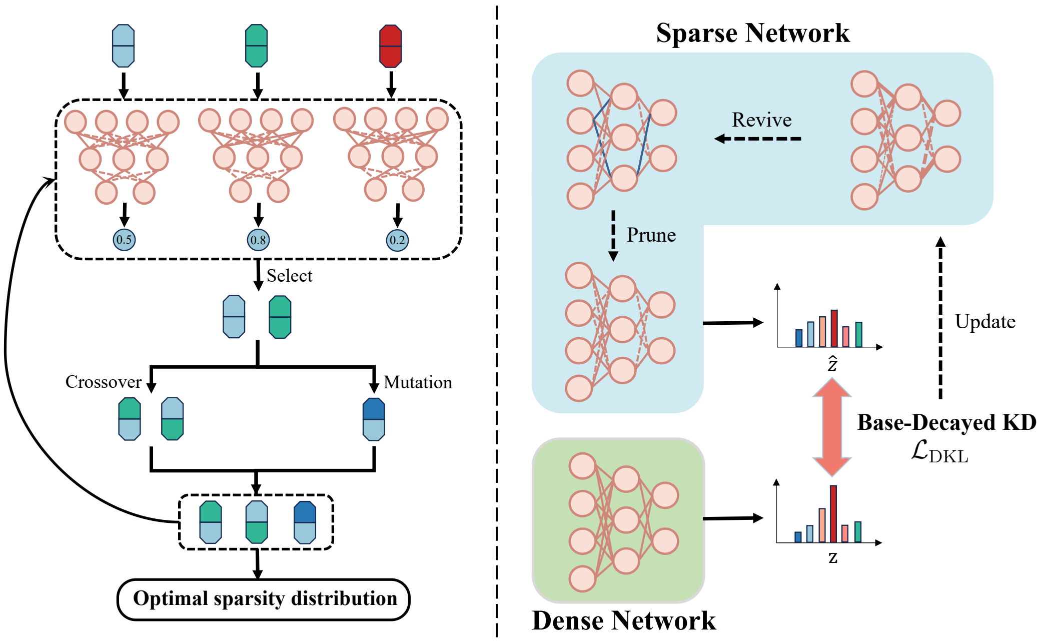

Figure 2. Overview about our method. **Left:**An overview of our method to search for sparsity distribution. We use evolutionary search to deal with the vast solution space. **Right:**We provide intuition for the training process. After finding optimal sparsity distribution, we use base-decayed KD loss and dynamic sparse training to retrain the pruned network.

and $\hat{\mathbf{Z}}$ from the sparse output, and propose a base-decayed sparsity objective. Kullback-Leibler divergence is used to quantify the difference between $\mathbf{Z}$ and $\hat{\mathbf{Z}}$:

$$\mathcal{L}_{\text{KL}} = \sum_{j=1}^{C} \mathbf{Z}_j \log_e \mathbf{Z}_j / \hat{\mathbf{Z}}_j, \tag{2}$$

where C represents the total number of target classes. Our principal impetus is to calibrate knowledge density transferred from the dense network to the sparse network, so as to avoid a sudden performance collapse at high sparsity rates. From Eq. (2), it is apparent that the intensity of supervision correlates to the base of log operation. We are deep in thought for the case of a high sparsity rate: there has a substantial gap between the dense and sparse outputs in the early training stage, propelling training instability; as training proceeds, learning from the dense network results in a rapid decline of the KL loss, which however, brings about a very small gradient and further insufficient training. Consequently, it is challenging to recover from the optimizing after-sparsified weights. Therefore, we propose to decay the base of log operation throughout the training process to adapt the loss scale as:

$$\mathcal{L}_{\text{DKL}} = \sum_{j=1}^{C} \mathbf{Z}_j \log_{e \cdot \gamma^t} \mathbf{Z}_j / \hat{\mathbf{Z}}_j, \tag{3}$$

where $\gamma < 1$ represents the decay rate and t is the current training epochs.

As the base decreases, our base-decayed sparsity objective appropriately attenuates and amplifies loss scale during training. Therefore, it promotes an efficient knowledge transferring from dense to sparse networks. It is important to highlight that our sparsity objective is significantly more efficient compared to the layer-wise MSE, as it fine-tunes the sparse networks in a global manner, eliminating the necessity for iterative retraining of each layer.

3.2.2 Reducing-Regrowing Sparsity Distribution

We proceed with the sparsity distribution issue. Current cutting-edge methods mainly hinge on an automatic differentiable training [17, 43, 46], which, however, becomes infeasible in the data-limited PTS. Given this, we resort to exploiting evolutionary algorithms, efficacy of which has been well verified in structured network sparsity [22, 26].

Denote $\mathbf{W} = \{\mathbf{W}^l\}_{l=1}^{L}$ as L-layer pre-trained model with weights and $\mathbf{R} = \{r^l\}_{l=1}^{L}$ as the sparsity distribution candidate where r^l signifies the sparsity rate of the l-th layer. Generally, as displayed in Figure 2, an evolutionary algorithm generally initiates a set of individuals, each of which possesses a fitness score to measure the performance of its sparsity distribution. The core of evolutionary algorithm is then to evolve these individuals from a series of crossover and mutation operations, and finally to pick up the one with best fitness. Algorithm 1 gives a comprehensive description of how to calculate the fitness. Despite the success of these methods in structured sparsity [20, 22], a

Algorithm 1 Fitness Calculation.

```
Input: $L$-layer pre-trained model with weights $\mathbf{W} = \{\mathbf{W}^l\}_{l=1}^L$; Sparsity candidate $\mathbf{R} = \{r^l\}_{l=1}^L$.; Calibration set $\mathcal{D}$; Global sparsity rate $P$; Excessive sparsity rate $P_e$;
Output: Fitness of candidate $\mathbf{R}$ ;
1:  ▷ Model Sparsification.
2:  Residual = $(P_e - P) \times numel(\mathbf{W})$ ▶ excessively sparsifying weights.
3:  $\mathbf{T}$ = softmax($\mathbf{R}$) $\times$ Residual    ▶ regrow weights.
4:  Sparsity = $\{P_e - \mathbf{T}^l / numel(\mathbf{W}^l)\}$
5:  $\hat{\mathbf{W}}$ = $Prune(\mathbf{W}$, Sparsity)
6:  ▷ Calibrate BN Statistics with Noisy Samples.
7:  $\mu, \sigma \leftarrow 0$    ▶ reset statistics.
8:  for Batch_Data in $\mathcal{D}$ do
9:      Noisy_Batch_Data = Batch_Data + Gaussian_Noise
10:     output = $\hat{\mathbf{W}}$(batch)
11: end for
12: ▷ Evaluate Calibration Accuracy as Fitness.
13: fitness $\leftarrow$ validate($\mathcal{D}$,$\hat{\mathbf{W}}$)
14: return fitness;
```

direct application in PTS encounters two dilemmas, as we analyze and address our uniques below.

First, colossal search space results from sparsifying weights, posing a challenge to ensure that the distribution candidate meets the desired global sparsity P. To address this, we propose a reducing-and-regrowing sparsification method in Lines 2 – 8 of Algorithm 1 to obtain the sparse model $\hat{\mathbf{W}}$. Specifically, we introduce another sparsification $P_e > P$ at every layer so as to reduce the search space first. Then, we further regrow the excessive sparsified weights using the evolutionary algorithm at a rate of $P_e - P$.

Second, assessing the fitness of a sparsity distribution candidate is also challenging, as evaluations easily lead to overfitting of the optimal structure to the calibration data, raising a difficulty to assess the fitness of a distribution candidate. Inspired by EagleEye [20], as shown in Lines 10–16 of Algorithm 1, for a sparse model with specific sparsity candidate, we recalculate the mean μ and variance σ of batch normalization layers on the calibration set $\mathcal{D}$, and then choose the performance of the sparse network on the calibration set as the fitness vale of the sparsity candidate. To avoid the overfitting risk, we apply random Gaussian noise to the input samples prior to evaluating fitness.

3.2.3 Sparsity Training

The training of our UniPTS is to derive the final status of the binary mask $\mathbf{M}$ called sparsity structure under the constraints of the proposed: 1) base-decayed sparsity objective; 2) reducing-regrowing sparsity distribution.

Predominant DST methods leverage manually designated metrics like magnitude of weight gradients [8] to proceed weight pruning and regrowing at intervals. However, within the scope of PTS, gradients lack reliability when determining essential weights, a consequence instigated by the limited quality and quantity of accessible data. We therefore consider the weight magnitude as the metric for pruning and regrowing. Given weights of the l-th layer $\mathbf{W}^l$ and the corresponding sparsity rate r^l, the mask $\mathbf{M}^l$ during training can be derived as:

$$\mathbf{M}^l_{i,j} = \begin{cases} 1 & \text{if } |\mathbf{W}^l_{i,j}| > \text{TopK}(|\mathbf{W}^l|, \lfloor (1 - r^l) \times S \rfloor), \\ 0 & otherwise, \end{cases} \tag{4}$$

where $|\cdot|$ is the absolute function and $\lfloor \cdot \rfloor$ is the floor operation. S = numel($\mathbf{W}^l$) is the number of parameters and TopK($|\mathbf{W}^l|, \lfloor(1 - r^l) \times S\rfloor$) returns $\lfloor(1 - r^l) \times S\rfloor$ largest value within $|\mathbf{W}^l|$. Then, the forward output $\hat{\mathbf{Y}}^l$ is derived as:

$$\hat{\mathbf{Y}}^l = (\mathbf{W}^l \odot \mathbf{M}^l)\hat{\mathbf{Y}}^{l-1}, \tag{5}$$

where $\odot$ denotes the Hadamard product.

Meanwhile, conventional DST methods commonly employ a periodic update sparsity structure, which means $\mathbf{M}$ will be updated every ΔT training iterations. For PTS, we propose to use an iteration-wise sparse training strategy *i.e.*, $\Delta T = 1$, as a periodic update may result in overfitting to a small amount of data. During training, we employ the Straight-Through Estimator [1] to approximate the gradient of pruned and unpruned weights during the backpropagation In this manner, the pruned weights are not reliant on the magnitude reduction of unpruned weights but can potentially recover through their own gradient, since both pruned and unpruned weights are subject to gradient updates.

Although such dynamic training enhances the diversity of sparse structures, it also aggravates fluctuation. Following [45], we modify the parameter update formula to prevent excessive fluctuation of the sparse structure and mitigate training instability issues. Specifically, a weight update mechanism is used to decay the magnitude of pruned weights by a certain proportion as:

$$\mathbf{W}^{l,t+1}_{i,j} = \begin{cases} \mathbf{W}^{l,t}_{i,j} - \beta * \dfrac{\partial \mathcal{L}}{\partial \mathbf{W}^{l,t}_{i,j}}, & \text{if } |\mathbf{W}^{l,t}_{i,j}| > \\ & \text{TopK}(|\mathbf{W}^{l,t}|, \lfloor (1 - r^l) \times S \rfloor), \\ \mathbf{W}^{l,t}_{i,j} - \beta * \dfrac{\partial \mathcal{L}}{\partial \mathbf{W}^{l,t}_{i,j}} - \alpha * \mathbf{W}^{l,t}_{i,j}, & otherwise, \end{cases} \tag{6}$$

where t denotes iterations, β represents the learning rate, and α is the decay proportion. This adjustment can play a role in controlling the variation of the sparse structure by limiting the magnitude of the pruned weight. Such that, our sparse training process for PTS can explore sufficient sparse structures while maintaining training stability.

Table 1. Image classification results on ImageNet-1K.

Model	Dense Top-1 Accuracy (%)	Sparsity Rate (%)	Top-1 Accuracy (%)			
			POT	RigL	STR	UniPTS
ResNet-18	69.76	50	69.22	69.18	33.70	**69.30**
		60	68.31	68.64	32.59	**68.51**
		70	65.91	67.46	30.52	**68.01**
		80	56.21	65.26	28.58	**66.35**
		90	14.15	58.17	25.88	**61.47**
ResNet-50	76.12	50	75.69	73.92	53.22	**75.76**
		60	74.36	73.20	50.10	**75.37**
		70	69.96	71.87	49.22	**74.73**
		80	48.04	69.33	48.32	**73.10**
		90	3.90	61.68	30.24	**68.60**
MobileNet-V2	72.05	50	69.25	66.57	30.96	**69.80**
		60	63.39	64.75	20.57	**68.01**
		70	47.07	60.72	14.62	**64.93**
		80	9.13	52.19	9.40	**59.47**
		90	0.2	30.44	5.32	**42.46**

4. Experiments

4.1. Settings

Datasets and networks. For ease of comparison, our experiments follow POT [18] which included image classification using CNNs on the ImageNet-1K [5] and object detection using Faster-RCNN [34] and SSD [25] on PASCAL VOC [9]. For image classification, we engage 10240 images from the ImageNet-1K training set to train sparse ResNet-18, ResNet-50 [13] and MobileNet-V2 [35]. For ResNet-18/50, the initial learning rate is 0.01 and the weight decay is 1×10^{-4}. For MobileNet-V2, the initial learning rate is 0.05 and the weight decay is 4×10^{-5}. For object detection, we use VGGNet-16 [37] as the backbone for Faster-RCNN and MobileNet-V1 [16] for SSD. We randomly select 10240 images from the training set which is composed of VOC2007 training set and VOC2012 training set. The stochastic gradient descent (SGD) optimizer is leveraged for training sparse networks with 16000 iterations. Uniformly across all networks, we implement a batch size of 64 and a cosine learning rate routine [27].

Implementation details. We implement UniPTS using Pytorch [32] and all experiments are conducted on a single NVIDIA RTX 3090. In particular, for the base-decayed sparsity objective in Eq. (3), we set the decay rate γ to 0.99. For the reducing-regrowing sparsity distribution, the excessive sparsity rate P_e is set as $(P+5)\%$. For sparsity training in Eq. (6), the decay proportion α is set as 3×10^{-5}.

Baselines. We evaluate our proposed UniPTS in comparison to the sole established PTS technique, POT [18]. In addition, we also compare the results of RigL [8], a classical sparse training method, and STR [17], a differentiable search method for sparsity distribution, in the context of post-training implementation, to provide a more exhaustive comparison.

4.2. Main Results

Image classification. Firstly, we report the quantitative results of various methods for pruning networks on image classification task, as outlined in Table 1. Comparatively, UniPTS consistently outperforms POT across all settings, with the accuracy gap widening as the sparsity rate increases. For instance, UniPTS is capable of boosting the accuracy of POT from a mere 3.9% to 68.60%, and that of MobileNet-V2 from 0.2% to 42.46% at a sparsity rate of 90%. Moreover, we contrast UniPTS with conventional sparsity methods on PTS scenario. Predicated upon our Post-training design, UniPTS consistently surpasses traditional methods across all sparsity rates. The poor performance of STR underscore the fact that traditional differentiable training for sparsity distribution is impractical in data-limited PTS scenarios, thereby supporting our design for reducing-regrowing sparsity distribution. Furthermore, results from RigL reveal the significance of dynamic sparse training for PTS at high sparsity rates, reinforcing our design for sparse training.

Object Detection. Moving beyond fundamental image classification benchmarks, we exploit the generalization capacity of UniPTS within the object detection task. Table 2 compares our proposed UniPTS to POT for pruning Faster-RCNN [34] and SSD [25] on PASCAL VOC [9] at 90% sparsity. Notably, UniPTS yields robust performance improvement of 3.3 and 3.4 mAP for pruning Faster-RCNN [34] and SSD [25], respectively. Given these favorable outcomes, the robustness and efficacy of UniPTS in object detection tasks are incontrovertibly confirmed.

Table 2. Object detection results at 90% sparsity rate.

Model	Method	mAP
Faster-RCNN	POT	51.29
Faster-RCNN	UniPTS	**54.59**
SSD	POT	57.02
SSD	UniPTS	**64.42**

N:M sparsity. In light of the likely demand for practical acceleration, we also appraise performance on the recently developed N:M semi-structured sparsity [44, 45], which stipulates at most N non-zero components within M consecutive weights to achieve expeditious inference aided by the N:M sparse tensor core [31]. The comparisons between UniPTS and POT on 2:4, 4:8 and 2:8 sparsity patterns are depicted in Table 3. It is evident that UniPTS can adeptly adapt to structured N:M sparsity. Regardless of the sparsity pattern, our method persistently supersedes POT by a noticeable margin.

Table 3. Comparison between POT and UniPTS for N:M sparsity.

		Sparse Pattern		
Model	Method	2:4	4:8	2:8
ResNet-18	POT	66.52	67.33	54.43
	UniPTS	**67.86**	**68.30**	**63.98**
ResNet-50	POT	73.64	74.53	47.24
	UniPTS	**74.83**	**75.14**	**71.46**
MobileNet-V2	POT	67.02	67.53	9.02
	UniPTS	**68.78**	**69.17**	**58.36**

Pruning efficiency. Moreover, we assess the pruning efficacy contrasting our proposed UniPTS and POT, delineated in Table 4. Compared to POT, UniPTS obviously holds an absolute advantage in the trade-off between pruning speed and accuracy. It is also worth noting that supplanting the search phase for sparsity distribution with the immediate application of the ERK budget [8] attenuates pruning time to an exponential magnitude. Though this precipitates a modest decline in accuracy, it engenders an auxiliary alternative for users navigating scenarios characterized by resource scarcity.

4.3. Ablation Study

In this section, we investigate the efficacy of each component in our method. To better understand the impact of these components on the overall performance, we conduct ablation experiments by replacing each component individually and show the performance on ImageNet-1K.

Sparsity objective. We first investigate the effect of our proposed base-decayed sparsity objective. In Table 5, we

Table 4. Time cost for pruning at 90% sparsity on ImageNet-1K. UniPTS* use ERK instead of searching sparsity distribution.

Model	Method	Time cost (min)	Top-1 Accuracy(%)
ResNet-18	POT	140	14.15
	UniPTS*	25	61.24
	UniPTS	260	61.47
ResNet-50	POT	484	3.90
	UniPTS*	58	66.97
	UniPTS	317	68.60
MobileNet-V2	POT	536	0.2
	UniPTS*	96	40.59
	UniPTS	357	42.46

examine the performance under five object variants including: 1) global MSE: instead of layer-wise MSE like POT, we try global MSE between $\mathbf{Z}$ and $\hat{\mathbf{Z}}$. 2) cross entory (CE): we use task relevant loss as sparsity objective and fine-tune the sparse network. 3) normal KL: we calculate the loss based on Eq. (2); 4) dynamic temperature: we introduce a dynamic temperature to smooth prediction probability; 5) dynamic base: we calculate the loss based on Eq. (3); As can be observed, our proposed base-decayed sparsity objective far surpasses other variants.

Table 5. Effect of sparsity objective when pruning ResNet-50 on ImageNet-1K.

Model	Strategy	Top-1 Accuracy (%)
ResNet-50	global MSE	13.07
ResNet-50	CE	38.99
ResNet-50	normal KL	64.47
ResNet-50	dynamic temperature	64.54
ResNet-50	dynamic base	**68.64**

Sparsity distribution. To assess the effect of our proposed sparsity distribution search, we execute a comparative study involving alternative methods for determining the sparsity distribution. These include heuristic design based on ERK [8] and the learnable sparsity distribution derived from STR [17]. Table 6 illustrates that our searched sparsity distribution yields the highest performance, with the margin of improvement escalating as the sparsity rate increases. In addition, we extend this searched sparsity distribution to POT. As shown in Table 7, the searched sparsity distribution can also contribute to improved performance in POT. As such, the effectiveness of our proposed sparsity distribution searching is validated. We visualize the sparsity distribution across different layers obtained by each method, as shown in Fig. 3. Notably, the learnable method proved to be ineffective, as it merely maintained a nearly uniform sparsity rate across all layers. In contrast, UniPTS effectively identified elegant layer-wise spar-

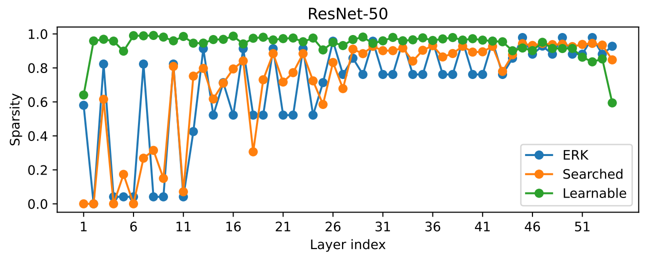

Figure 3. Sparsity distribution obtained by different methods.

sity rates (*e.g.*, preserving more weights for the initial layers), thereby demonstrating its superior capability.

Table 6. Effect of sparsity distribution when pruning ResNet-50 on ImageNet-1K using UniPTS.

Model	Method	Sparsity(%)	Top-1 Accuracy (%)
ResNet-50	ERK	90.00	66.97
ResNet-50	Learnable	89.75	64.48
ResNet-50	Searched	90.15	**68.60**
ResNet-50	ERK	80.00	71.53
ResNet-50	Learnable	79.94	71.86
ResNet-50	Searched	80.13	**73.10**
ResNet-50	ERK	70.00	73.44
ResNet-50	Learnable	69.95	74.19
ResNet-50	Searched	70.17	**74.74**

Table 7. Effect of sparsity distribution when pruning ResNet-50 on ImageNet-1K using POT.

Model	Method	Sparsity(%)	Top-1 Accuracy (%)
ResNet-50	L2-norm	90.00	3.88
ResNet-50	Searched	90.15	**5.07**
ResNet-50	L2-norm	80.00	48.07
ResNet-50	Searched	80.13	**50.85**
ResNet-50	L2-norm	70.00	69.95
ResNet-50	Searched	70.17	**70.92**

Sparsity Training. For sparsity training, we first investigate the effect of the decay factor α in Eq. (6). Figure 4 plots accuracy of pruned ResNet-50 at a sparsity rate of 90% with different α adopted. It is intuitive that α indicates the degree of pruned weight decay, where $\alpha = 0$ indicates that pruned weights are updated exactly according to their gradient, and $\alpha \neq 0$ indicates pruned weights will decay to avoid reviving. With α increasing, the accuracy increases first and then decreases. A large α will lead to limited sparse structures. And a suitable α can constrain fluctuation and improve the performance.

Furthermore, investigate the influence of the update interval ΔT for the pruning masks. Figure 4 shows the performance of UniPTS under different intervals $\Delta T \in [1, 10, 100, 1000]$. As can be observed, the best accuracy is

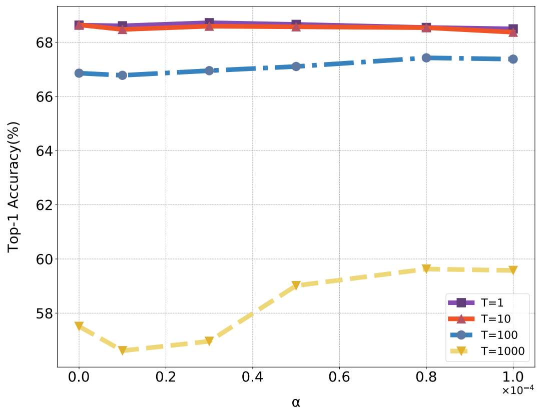

Figure 4. Performance influence of the sparse training strategy.

consistently obtained when the pruning mask is updated every iteration, *i.e.*, $\Delta T = 1$. Furthermore, as ΔT increases, performance gradually declines. For instance, the accuracy drops from 66.86% to 57.51% when α equals 0 and the interval increases from 100 to 1000. This phenomenon appears to be in contrast with traditional practice in DST, where a larger interval yields better accuracy. Nonetheless, in the context of PTS, longer intervals imply that the learned mask will depend on the entirety of the validation set, leading to overfitting.

5. Conclusion

In this paper, we focus on mitigating the performance gap between PTS and traditional network sparsity. We advance UniPTS, a unified framework comprising a base-decayed sparsity objective, a reducing-regrowing sparsity distribution, and a dynamic sparse training on the basis of the two preceding aspects to optimize the sparsity structure. Extensive experiments across a panoply of computer vision tasks demonstrate the effectiveness of UniPTS, which surpasses prior works by a significant margin, especially at high sparsity rates. Our work engenders new insights into enhancing the performance of PTS and we hope to stimulate further research into exploring the benefits of network sparsity from a more pragmatic perspective.

Acknowledgement

This work was supported by National Science and Technology Major Project (No. 2022ZD0118202), the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No. U21B2037, No. U22B2051, No. 62176222, No. 62176223, No. 62176226, No. 62072386, No. 62072387, No. 62072389, No. 62002305 and No. 62272401), and the Natural Science Foundation of Fujian Province of China (No.2021J01002, No.2022J06001).

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 5

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1877–1901, 2020. 1

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 43(5):1483–1498, 2019. 1

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 6

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[7] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 1

[8] 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 (ICML)*, pages 2943–2952, 2020. 1, 2, 3, 5, 6, 7

[9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88:303–338, 2010. 6

[10] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph H Hassoun. Post-training piecewise linear quantization for deep neural networks. In *European Conference on Computer Vision (ECCV)*, pages 69–86, 2020. 1, 2

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 2, 3

[12] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015. 1, 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 6

[14] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, pages 784–800, 2018. 3

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 6

[17] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning (ICML)*, pages 5544–5555, 2020. 2, 3, 4, 6, 7

[18] Ivan Lazarevich, Alexander Kozlov, and Nikita Malinin. Post-training deep neural network pruning via layer-wise calibration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 798–805, 2021. 1, 2, 3, 6

[19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018. 2, 3

[20] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *European Conference on Computer Vision (ECCV)*, pages 639–654, 2020. 4, 5

[21] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. 2

[22] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. *arXiv preprint arXiv:2001.08565*, 2020. 4

[23] Junjie Liu, Zhe Xu, Runbin Shi, Ray CC Cheung, and Hayden KH So. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. *arXiv preprint arXiv:2005.06870*, 2020. 2, 3

[24] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:9908–9922, 2021. 2, 3

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016. 6

[26] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3305, 2019. 3, 4

[27] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017. 6

[28] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction, 2019. 2

[29] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning (ICML)*, pages 7197–7206, 2020. 1, 2

[30] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang. Dsa: More efficient budgeted pruning via differentiable sparsity allocation. In *European Conference on Computer Vision (ECCV)*, pages 592–607, 2020. 3

[31] Nvidia. Nvidia a100 tensor core gpu architecture. `https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf`, 2020. 7

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8026–8037, 2019. 6

[33] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 1, 2

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 6

[35] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 6

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Machine Learning (ICML)*, 2015. 6

[38] Kai Sheng Tai, Taipeng Tian, and Ser Nam Lim. Spartan: Differentiable sparsity via regularized transportation. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:4189–4202, 2022. 3

[39] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020. 2

[40] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*, 2022. 2

[41] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European Conference on Computer Vision (ECCV)*, pages 68–85, 2022. 1

[42] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017. 1

[43] Yuxin Zhang, Mingbao Lin, Chia-Wen Lin, Jie Chen, Yongjian Wu, Yonghong Tian, and Rongrong Ji. Carrying out cnn channel pruning in a white box. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2022. 4

[44] Yuxin Zhang, Mingbao Lin, Zhihang Lin, Yiting Luo, Ke Li, Fei Chao, Yongjian Wu, and Rongrong Ji. Learning best combination for efficient N:M sparsity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 7

[45] Aojun Zhou, Junnan Zhu Yukun Ma, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations (ICLR)*, 2021. 5, 7

[46] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3599–3608, 2021. 2, 3, 4