MaxQ: Multi-Axis Query for N:M Sparsity Network

Jingyang Xiang[1] Siqi Li[1] Junhao Chen[1] Zhuangzhi Chen[2] Tianxin Huang[1]
Linpeng Peng[1] Yong Liu[1*]
[1]APRIL Lab, Zhejiang University, Hangzhou, China
[2]IVSN, Zhejiang University of Technology, Hangzhou, China

Abstract

N:M sparsity has received increasing attention due to its remarkable performance and latency trade-off compared with structured and unstructured sparsity. However, existing N:M sparsity methods do not differentiate the relative importance of weights among blocks and leave important weights underappreciated. Besides, they directly apply N:M sparsity to the whole network, which will cause severe information loss. Thus, they are still sub-optimal. In this paper, we propose an efficient and effective Multi-Axis Query methodology, dubbed as MaxQ, to rectify these problems. During the training, MaxQ employs a dynamic approach to generate soft N:M masks, considering the weight importance across multiple axes. This method enhances the weights with more importance and ensures more effective updates. Meanwhile, a sparsity strategy that gradually increases the percentage of N:M weight blocks is applied, which allows the network to heal from the pruning-induced damage progressively. During the runtime, the N:M soft masks can be precomputed as constants and folded into weights without causing any distortion to the sparse pattern and incurring additional computational overhead. Comprehensive experiments demonstrate that MaxQ achieves consistent improvements across diverse CNN architectures in various computer vision tasks, including image classification, object detection and instance segmentation. For ResNet50 with 1:16 sparse pattern, MaxQ can achieve 74.6% top-1 accuracy on ImageNet and improve by over 2.8% over the state-of-the-art. Codes and checkpoints are available at https://github.com/JingyangXiang/MaxQ.

1. Introduction

Deep convolutional neural networks (CNNs) have achieved great success in various computer vision tasks, including image classification [17, 39], object detection [12, 18], semantic segmentation [7, 13]. However, the expensive memory and computational overhead have presented challenges for deploying them on mobile or edge devices. Therefore, it is vital to study the network compression to reduce its runtime overhead while maximally retaining its performance.

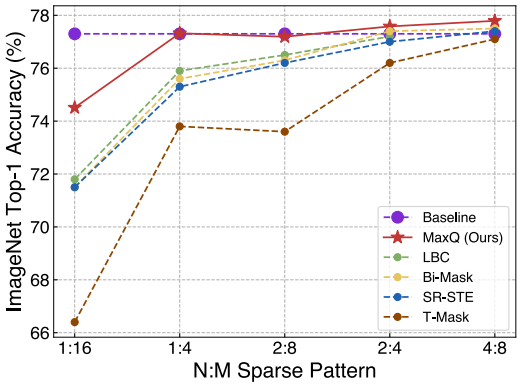

Figure 1. Comparison of the accuracy-sparse pattern Pareto curve of the ResNet50 on ImageNet. MaxQ shows the top-performing Pareto frontier compared with previous N:M sparsity methods [1, 25, 44, 45].

Among the many compression methods [15, 22, 24, 33, 37], network sparsity stands out as a highly effective tool for achieving practical memory and FLOPs saving. Most researchers have paid their attention to structured sparsity [20, 21, 30] and unstructured sparsity [16, 28]. However, both of them have certain shortcomings in terms of performance and acceleration, limiting their application. How to realize a better trade-off remains an open problem, as it hinges on factors like the granularity of sparsity and hardware design.

Recently, N:M sparsity [35], a fine-grained sparsity pattern, is considered a highly promising solution. It can achieve a better trade-off between performance and latency compared to structured and unstructured sparsity thanks to its hardware and software co-design [29, 35]. Several studies have enhanced the performance of N:M sparsity. The pioneer work ASP [35] proposes to remove the two weights

*Corresponding author

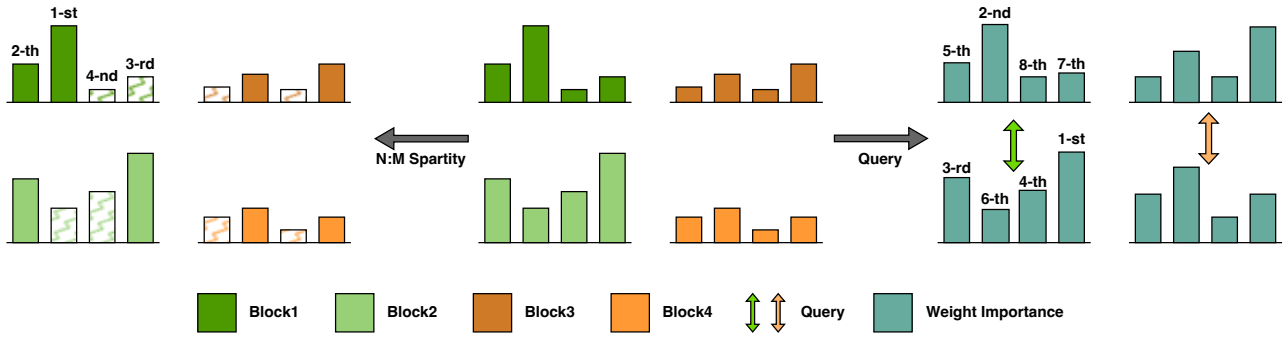

Figure 2. The framework of our MaxQ method, which queries the weights importance among the blocks and generates soft masks by querying the weight across multiple axes. For simplify, we only show single axis query.

with the smallest magnitude from every four consecutive weights and use the pretrain-prune-finetune (PPF) pipeline to get the sparse network. This approach effectively preserves the performance of the dense network while it still suffers from expensive computation. To address this, Zhou *et al*. [1] proposes to learn N:M sparsity from scratch by extending a regularization term to Straight-Through Estimator [3], which improves the efficiency of sparse architecture update. Zhang *et al*. [44] treats the N:M sparsity problem as a combinatorial problem and solves it in an efficient divide-and-conquer manner. Although these methods improve the training efficiency or performance of N:M sparsity network, they are typically constructed upon the premise of weight block independence and do not differentiate the weight importance among blocks. At the same time, these methods apply N:M sparsity to the whole network N:M from the beginning of training, leading to critical information loss. As a result, they are still sub-optimal.

In this paper, we propose a simple yet effective *Multi-Axis Query* methodology, dubbed as **MaxQ**, to identify the weight importance and build high-performance N:M sparsity network. In contrast to the previous approaches, which considered the weights in N:M blocks individually, MaxQ queries the weight importance along multiple axes to identify more critical connections and ensures more effective updates for them. It is worth saying MaxQ achieves this according to the threshold from weight magnitude via a dynamic and parameter-free approach, which is different from the previous methods achieving this by learning a threshold parameter. Therefore, it requires no additional learnable parameters and simplifies the training process. Additionally, MaxQ follows an incremental pruning schedule, which gradually increases the percentage of N:M sparse block based on the epoch. It progressively allows the network to heal from the pruning-induced damage and notably enhances performance. This strategy makes the training process more stable and weights to be trained more sufficiently, thus improving convergence and performance. With only one training pass from scratch, our obtained sub-models perform better than the previous method across diverse CNN architectures in various computer vision tasks. Moreover, our MaxQ can achieve good results in Post-Training-Quantization (PTQ) even though it is a self-structured re-parameterized network, contrary to the previous structured re-parameterized network.

The contributions of our work are highlighted as follows:

- We propose a multi-axis query method MaxQ with two novel features: (1) a multi-axis query approach to identify important connections among the N:M sparse blocks; (2) a dynamic approach to generate soft pruning masks in a parameter-free manner.
- Our method follows an incremental pruning schedule by increasing the percentage of N:M blocks according to their ℓ_1-norm. It enhances the performance of the N:M sparsity network in one training pass.
- Experimentally, MaxQ achieves consistent improvement across different N:M sparse patterns compared to previous methods (as shown in Fig. 1) and is applicable to downstream tasks for computer vision. For image classification on ImageNet, MaxQ achieves 74.6% top-1 accuracy for a 1:16 sparse ResNet50, improving the previous best [44] by 2.8%. For object detection and instance segmentation on the COCO [32] dataset, MaxQ can achieve comparable results with a dense baseline model under the 1:4 structured sparsity.
- MaxQ is friendly to quantization. The ResNet50 with 2:4 sparsity achieves 0.5% drop (77.6%→77.1%) in the top-1 accuracy when quantized to INT8 using PTQ, which benefits its deployment.

2. Related Work

2.1. Sparsity Granularity in Network Compression

Sparsity is an essential tool for network compression and has attracted extensive attention from both industry and academia. According to the granularity of sparse weights, network sparsity can be categorized into structured sparsity [19, 30, 41], unstructured sparsity [10, 11, 14, 16, 27,

Granularity	Performance	Latency	Hardware	Library
Filter	✓	✓✓✓	General	General
Weight	✓✓✓	✓	Specific	Specific
1×N	✓✓	✓✓	General	Specific
N:M	✓✓	✓✓✓	Ampere GPUs	General

Table 1. An overview of mainstream sparsity granularity.

28] and semi-structured sparsity [5, 9, 31, 35]. Tab. 1 overviews these sparsity and their characteristics. Structured sparsity removes the entire channel of a convolutional or fully connected layer, which enables significant speedup on general-purpose hardware. Unstructured sparsity removes single weight and achieves negligible performance loss even under a large sparse ratio [10, 27]. However, the former suffers huge performance degradation at high compression ratios, and the latter gains rare speedup on general-purpose hardware for its expensive memory access and low computational density resulting from irregular sparse tensors. In recent years, semi-structured sparsity has received extensive attention from researchers. By limiting the distribution of weights to specific sparse types, semi-structured sparsity can achieve a better trade-off between performance and latency through a synergistic software and hardware design. Among the semi-structured sparsity, 1×N [9, 31] and N:M [35] are now widely supported by software and hardware. 1×N makes N consecutive output channels keep the same sparse pattern, enabling significant speedup on general-purpose hardware. N:M sparsity forces at most N of the M consecutive weights along the input channel dimension to be non-zero, achieving good results while maintaining high sparsity and high-performance acceleration on NVIDIA Ampere GPUs.

The core purpose of network sparsity is to remove unimportant weights from the network and preserve the network's performance. Therefore, learning the appropriate mask is indeed necessary. Mask is usually obtained from the weight values according to specific criteria, and the process must follow a schedule. We empirically categorize existing studies into two groups below based on criteria and schedule.

Mask Criteria. Extensive studies have investigated how to identify the critical weights in a network. Among them, the most widely used is the magnitude-based criterion that selects the pruning targets by their absolute value or ℓ_1-norm. Apart from this, He *et al*. [21] proposed to prune filters via geometric median to compress CNN models with redundancy rather than those with "relatively less" importance. Molchanov *et al*. [36] proposed to estimate the contribution of a neuron (filter) to the final loss via the first and second-order Taylor expansions. Lee *et al*. [28] introduced a saliency criterion based on connection sensitivity to identify structurally important connections in the network. Some other criteria, such as BN-based [34], have also proven effective.

Mask Schedule. Learning schedule for masks is also vital for a sparse network. Most of the early research [16, 30] follows the PPF pipeline iteratively, which is complicated to use and time-consuming. In order to simplify this process, recent studies use soft and incremental pruning to train sparse networks from scratch without dependence on pre-trained weights. For example, Humble *et al*. [26] proposed soft masking for channel pruning, which allows pruned channels to adaptively return to the network while simultaneously pruning towards a target cost constraint. Hou *et al*. [23] proposed a channel exploration methodology to prune and regrow the channels with a sampling-based strategy repeatedly. Evci [10] introduced a sparse-to-sparse training procedure with a fixed parameter count and a fixed computational cost, sacrificing no accuracy relative to existing dense-to-sparse training methods. Compared to hard and one-shot pruning, these soft and incremental strategies can improve the model's performance, especially at a high sparse ratio.

3. Methodology

3.1. Preliminaries

Here, we define the N:M sparsity problem. Without loss of generality, we suppose an L-layer CNN with parameters $\mathbf{W} = \{\mathbf{w}^1, ..., \mathbf{w}^L\}$, with $\mathbf{w}^l \in \mathbb{R}^{C^l_{\text{out}} \times C^l_{\text{in}} \times K^l_{\text{h}} \times K^l_{\text{w}}}$. C^l_{out}, C^l_{in}, K^l_{h} and K^l_{w} represent the number of output channels, input channels, kernel height and kernel width for l-th layer respectively. N:M sparsity groups every M consecutive weights along the input channel in each layer and requires each group to have at most N non-zero elements. With this pattern, only N non-zero values in each group need to be stored and metadata is adopted to encode the position of each non-zero value. This process can compress the origin matrix and be accelerated by designated processing units (*e.g*. NVIDIA Ampere Tensor Cores). To this end, we first rearrange $\mathbf{w}^l$ to $\mathbf{m}^l \in \mathbb{R}^{G^l \times M}$, where $G^l = \frac{C^l_{\text{out}} \times K^l_{\text{h}} \times K^l_{\text{w}} \times C^l_{\text{in}}}{M}$ and denote $\mathbf{M} = \{\mathbf{m}^1, ..., \mathbf{m}^L\}$. Meanwhile, N:M binary masks $\mathbf{b}^l \in \{0, 1\}^{G^l \times M}$ are introduced to achieve such sparsity Then, the optimization objective can be formulated as:

$$\min_{\mathbf{M},\mathbf{B}} \mathcal{L}(\mathbf{M} \cdot \mathbf{B}; \mathcal{D}) \;\; s.t. \|\mathbf{b}^l_{g^l,:}\|_0 \leqslant N \tag{1}$$

where $g^l = 1, 2, ..., G^l$. The $\mathcal{L}(\cdot)$ represents the loss function and $\mathcal{D}$ denotes the observed data.

3.2. Multi-Axis Query

Although extensive studies have promoted the performance of N:M sparsity, they only focus on picking the N most important weights from M elements and applying binary masks to them, which makes a portion of the weights with

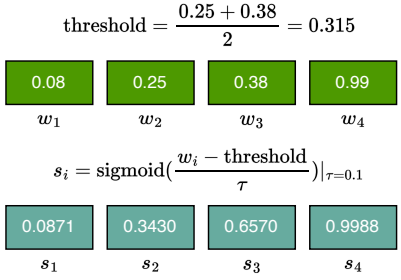

Figure 3. The process of computing soft masks. The weights of the model are sorted in descending order based on their magnitudes for clear understanding. We assume $(N, p, \tau) = (4, 0.5, 0.1)$.

more importance underappreciated. Therefore, it is vital to exploit the importance of weights further to improve the performance of N:M sparsity networks.

To represent the importance of the weights in the N:M sparse pattern, we first relax the constrain of the binary matrix $\mathbf{b}^l$ to $\mathbf{s}^l \in \{x | x \geqslant 0\}^{G^l \times M}$. Then, the optimization objective can be rewritten as:

$$\min_{\mathbf{M},\mathbf{S}} \mathcal{L}(\mathbf{M} \cdot \mathbf{S}; \mathcal{D}) \;\; s.t. \|\mathbf{s}^l_{g^l,:}\|_0 \leqslant N \tag{2}$$

It can be seen as replacing the previous hard mask $\mathbf{b}^l$ with a soft mask $\mathbf{s}^l$. $\mathbf{s}^l$ satisfies the N:M sparse pattern and can be folded into the network as constants, which will not cause any distortion to the sparse pattern and introduce any extra cost during the runtime.

To get the value of $\mathbf{s}^l$, we propose MaxQ to measure the importance of weights. As shown in Fig. 2, MaxQ can be divided into two parts in what follows.

Part 1: Apply N:M Sparsity.

First, for l-th layer, we initialize all elements in $\mathbf{b}^l$ to 1. For t-th epoch, we sort each element in $\mathbf{m}^l_g$ according to it absolute value and sort the $\mathbf{m}^l$ according to it's ℓ_1-norm to identify the set of blocks to apply N:M sparsity:

$$\begin{aligned} \mathcal{M}^l_g &= \text{ArgTopK}_{\text{M-N}}\left(-\left|\mathbf{m}^l_{g,:}\right|\right) \\ \mathcal{T}^l_t &= \text{ArgTopK}_{\lceil G^l \delta_t \rceil}\left(\left\{\left\|\mathbf{m}^l_g\right\|_1\right\}\right) \end{aligned} \tag{3}$$

where $\mathcal{M}^l \in \mathbb{R}^{G^l \times (M-N)}$, $\mathcal{T}^l_k \in \mathbb{R}^{\lceil G^l \delta_t \rceil}$ and δ_t represents the percentage of weight blocks to apply N:M sparsity, which gives the indices of weights with the $(M-N)$ smallest absolute value in each block and blocks with the top $\lceil G^l \delta_t \rceil$ value in ℓ_1-norm respectively. Then we prune the weight by zeroizing $\{\mathbf{b}^l_{i,j} | i \in \mathcal{T}^l_t, j \in \mathcal{M}^l_i\}$.

Part 2: Measure Importance. We show our function $S = G(V, p, \tau)$ to measure weight importance. Give a vector $V \in \mathbb{R}^N$ and a sparse rate p, we get the threshold σ by

$$\begin{cases} \sigma_h &= \min\left(\text{topK}\left(\text{abs}(V), (1-p) \cdot N\right)\right) \\ \sigma_l &= \max\left(\text{topK}\left(-\text{abs}(V), p \cdot N\right)\right) \\ \sigma &= \left(\text{abs}(\sigma_h) + \text{abs}(\sigma_l)\right)/2 \end{cases} \tag{4}$$

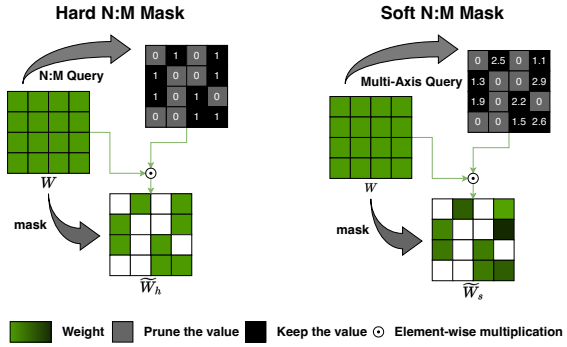

Figure 4. MaxQ to generate soft masks.

Then, we measure weight importance with sigmoid:

$$s_i = \text{sigmoid}\left(\left(|v_i| - \sigma\right)/\tau\right) \tag{5}$$

where $i = 1, 2, ..., N$ and τ is the global temperature parameter to control the level of softness in the masks. Fig. 3 shows the process when $(N, p, \tau) = (4, 0.5, 0.1)$.

To measure the weight importance among the blocks, we query the weight along filter axis and kernel axis to generate corresponding soft masks:

$$\begin{aligned} \mathbf{s}^{l^{(f)}}_{i,:,:,:} &= \left(G\left(w^l_{i,:,:,:}, (M-N)/M, \tau\right)\right) \\ \mathbf{s}^{l^{(k)}}_{:,:,k_1,k_2} &= \left(G\left(w^l_{:,:,k_1,k_2}, (M-N)/M, \tau\right)\right) \end{aligned} \tag{6}$$

where $i = 1, 2, ..., C^l_{\text{out}}$, $k_1 = 1, ..., K^l_h$ and $k_2 = 1, ..., K^l_w$.

We rearrange (RA) the $\mathbf{s}^{l^{(f)}}$ and $\mathbf{s}^{l^{(k)}}$ to match the shape of $\mathbf{b}^l$ and obtain the soft mask as shown in Fig. 4:

$$\begin{aligned} \mathbf{s}^l &= \mathbf{b}^l + \mathbf{b}^l \odot \text{RA}(\mathbf{s}^{l^{(f)}}) + \mathbf{b}^l \odot \text{RA}(\mathbf{s}^{l^{(k)}}) \\ &= \mathbf{b}^l \odot \left(1 + \text{RA}(\mathbf{s}^{l^{(f)}}) + \text{RA}(\mathbf{s}^{l^{(k)}})\right) \end{aligned} \tag{7}$$

It is worth noting $\mathbf{s}^l$ can be precomputed. Meanwhile, since $\mathbf{b}^l$ satisfies the N:M sparse pattern, $\mathbf{s}^l$ will also satisfy this pattern. Therefore, MaxQ will not bring any additional overhead during the runtime compared to the conventional N:M sparsity network.

3.3. Incremental Sparsity

Previous methods fixed δ_t to 1 throughout the training process, which means the whole network is in N:M sparse pattern from the beginning of training. It will cause severe information loss and is detrimental to network convergence.

To this end, incremental sparsity, which gradually increases the sparse ratio based on the current epoch/step, has been shown to be an effective technique to heal sparse networks from pruning and improve their performance. For unstructured sparsity, incremental sparsity can be achieved by gradually removing a single weight; for structured sparsity, it can be achieved by removing the channels across layers. Since networks often have a large number of parameters and

Algorithm 1: Overview of the MaxQ method.

Input: An L-layer CNN model with weights $\mathbf{W} = \{\mathbf{w}^1, ..., \mathbf{w}^L\}$; target sparse pattern N:M; total training epochs T_{total}; initial and final epoch for incremental pruning t_i, t_f; training set $\mathcal{D}$;
Output: A sub-model satisfying the target sparse pattern N, M and its optimal weight values $\mathbf{W}^*$;
Randomly initialize the model weights $\mathbf{W}$;
Rearrange the model weights $\mathbf{W}$ to $\mathbf{M}$;
for *each training epoch* $t \in [1, ..., T_{total}]$ **do**
  Compute the percentage of N:M blocks δ_t via Eq. (8);
  for *each mini-batch* $\in \mathcal{D}$ **do**
    for $l \in [1, ..., L]$ **do**
      Reset $\left\{\mathbf{b}^l_{i,j} | \forall i, \forall j\right\}$ to 1 ;
      Get the indices $\mathcal{M}^l$ and $\mathcal{T}^l_t$ via Eq. (3) ;
      Set $\left\{\mathbf{b}^l_{i,j} | i \in \mathcal{T}^l_t, j \in \mathcal{M}^l_i\right\}$ to 0 ;
      Get the $\mathbf{s}^{l(f)}$ and $\mathbf{s}^{l(k)}$ via Eq. (4) $\sim$ Eq. (6) ;
      Get the soft mask $\mathbf{s}^l$ via Eq. (7)
    Forward via Eq. (2) ;
    Backward and update via the SGD optimizer ;
Compute $\mathbf{M}^* = \{\mathbf{m}^l \cdot \mathbf{s}^l | \ \forall l \ \}$;
Rearrange $\mathbf{M}^*$ back to $\mathbf{W}^*$;

channels, the sparsity process can be viewed as continuous. Different from them, N:M sparse is characterized by several M consecutive weight groups. If we increase the sparsity by simply reducing $\left\|\mathbf{b}^l_{g,:}\right\|_0$ from M to N, the sparse networks will suffer critical performance degradation, especially at high sparsity.

In this paper, we propose gradually increasing the percentage of N:M sparse blocks to achieve a smoother sparsity process. We apply the same sparse schedule for each layer The percentage of N:M sparse blocks at the t-th training epoch is computed as:

$$\delta_t = \min(1, \max(0, 1 - [1 - (t - t_i)/(t_f - t_i)]^3)) \quad (8)$$

where t_i and t_f denote the beginning and ending epochs in the incremental sparsity process. Specifically, if t is smaller than t_i, the network is trained in a dense state, and when t is larger than t_f, the network is trained in a N:M sparse pattern. Notably, we firstly apply the N:M sparse pattern for blocks with larger ℓ_1-norm because we find it can reduce the performance degradation when the sparsity increases and keep the convergence process stable. We conduct ablation studies in Sec. 4.5 to show its advantage. As an algorithm guideline, the pseudo-code of MaxQ is provided in Algorithm 1.

4. Experiment

4.1. Experiment Settings

To validate the effectiveness of MaxQ, we conducted image classification on ImageNet with heaveweight CNNs ResNet34 [17], ResNet50 [17] and lightweight MobileNetV1 [24]. We compare MaxQ with N:M sparsity and unstructured sparsity in Sec. 4.2 and Sec. 4.3, respectively. We also conducted object detection and semantic segmentation on the COCO [32] benchmark with Faster-RCNN [38] and Mask-RCNN [18] in Sec. 4.4. All experiments are implemented on PyTorch with NVIDIA RTX 3090 and trained with the same configurations as previous work [1, 44] to make a fair comparison. t_i and t_f are set to 0 and 3/4 of the total training epochs respectively. Ablation studies about components, t_i and t_f and strategy for incremental sparsity are demonstrated in Sec. 4.5. Performance analysis is shown in Sec. 4.6.

Model	Method	N:M	Top-1	Epochs	FLOPs	Params
ResNet34	Baseline	-	74.6%	120	3.67G	21.8M
	ASP	1:4	70.9%	200	1.01G	5.85M
	SR-STE	1:4	73.8%	120	1.01G	5.85M
	LBC	1:4	73.7%	120	1.01G	5.85M
	MaxQ	1:4	**74.2%**	120	1.01G	5.85M
	ASP	2:4	73.9%	200	1.90G	11.2M
	SR-STE	2:4	74.3%	120	1.90G	11.2M
	LBC	2:4	74.1%	120	1.90G	11.2M
	MaxQ	2:4	**74.5%**	120	1.90G	11.2M
ResNet50	Baseline	-	77.3%	120	4.11G	25.6M
	ASP	2:4	77.4%	200	2.12G	13.8M
	SR-STE	2:4	77.0%	120	2.12G	13.8M
	LBC	2:4	77.2%	120	2.12G	13.8M
	MaxQ	2:4	**77.6%**	120	2.12G	13.8M
	ASP	1:4	76.5%	200	1.11G	7.93M
	SR-STE	1:4	75.3%	120	1.11G	7.93M
	LBC	1:4	75.9%	120	1.11G	7.93M
	MaxQ	1:4	**77.3%**	120	1.11G	7.93M
	ASP	2:8	76.6%	200	1.11G	7.93M
	SR-STE	2:8	76.2%	120	1.11G	7.93M
	LBC	2:8	76.5%	120	1.11G	7.93M
	MaxQ	2:8	**77.2%**	120	1.11G	7.93M
	ASP	1:16	71.5%	200	0.44G	3.52M
	SR-STE	1:16	71.5%	120	0.44G	3.52M
	LBC	1:16	71.8%	120	0.44G	3.52M
	MaxQ	1:16	**74.6%**	120	0.44G	3.52M

Table 2. Results of the different N:M sparsity training methods for ResNet34 and ResNet50 on ImageNet.

Model	Method	N:M	Top-1	Epochs	FLOPs	Params
MobileNetV1	Baseline	-	71.9%	120	578M	4.23M
	ASP	2:4	70.4%	200	302M	2.66M
	SR-STE	2:4	71.5%	120	302M	2.66M
	MaxQ	2:4	**72.1%**	120	302M	2.66M
	ASP	1:4	65.4%	200	164M	1.88M
	SR-STE	1:4	67.8%	120	164M	1.88M
	MaxQ	1:4	**68.5%**	120	164M	1.88M

Table 3. Results of the different N:M sparsity training methods for lightweight model MobileNetV1 on ImageNet.

4.2. Comparison with N:M sparsity

We first apply our MaxQ to ResNet34 and ResNet50 to validate its effectiveness. As shown in Tab. 2, MaxQ leads all N:M sparse patterns and networks. For ResNet34, MaxQ

Method	Top-1	Sparsity	FLOPs	Params	S	U
Baseline	77.3%	0.0	4.10G	25.6M	-	-
RigL [10]	74.6%	80	0.92G	5.12M	✗	✓
GMP [46]	75.6%	80	0.82G	5.12M	✗	✓
MAP [2]	75.9%	80	-	5.12M	✗	✗
STR [27]	76.2%	81	0.82G	5.12M	✗	✓
SR-STE	75.3%	1:4	1.13G	7.97M	✓	✓
LBC	75.9%	1:4	1.13G	7.97M	✓	✓
MaxQ	**77.3%**	1:4	1.13G	7.97M	✓	✓
SR-STE	76.2%	2:8	1.13G	7.97M	✓	✓
LBC	76.5%	2:8	1.13G	7.97M	✓	✓
MaxQ	**77.2%**	2:8	1.13G	7.97M	✓	✓
DNW [42]	68.3%	95	0.20G	1.28M	✗	✗
RigL [10]	70.0%	95	0.49G	1.28M	✗	✓
GMP [46]	70.6%	95	0.20G	1.28M	✗	✓
STR [27]	70.4%	95	0.16G	1.24M	✗	✗
OptG [43]	72.5%	95	0.22G	1.28M	✗	✗
SR-STE	71.5%	1:16	0.44G	3.52M	✓	✓
LBC	71.8%	1:16	0.44G	3.52M	✓	✓
MaxQ	**74.6%**	1:16	0.44G	3.52M	✓	✓

Table 4. Results of the N:M and unstructured sparsity methods for ResNet50 on ImageNet. S: Structured. U: Uniform.

outperforms SR-STE [1] and LBC [44] at 1:4 sparse pattern at the top-1 accuracy by 0.4% and 0.5% respectively. For ResNet50, MaxQ achieves similar improvement at 1:4 and 2:8 sparse patterns. Meanwhile, we also conduct experiments on lightweight CNN MobileNetV1, as compressing this lightweight network is more beneficial for further acceleration on mobile devices. Similar to ResNet, we apply N:M sparsity to all except the first, last layers and depthwise convolutional layers. The results in Tab. 3 also demonstrate the superiority against the others.

The results in Tab. 2 and Tab. 3 also show two interesting properties of our MaxQ method. ResNet34, ResNet50 and MobileNetV1 achieve 74.5%, 77.6% and 72.1% top-1 accuracy at 2:4 sparse pattern, while their dense counterpart achieves 74.6%, 77.3% and 71.9% respectively, which means our MaxQ can achieve almost lossless or better results on various networks at 2:4 sparse pattern. On the other hand, our MaxQ achieves a more significant improvement at a high sparse ratio. For example, our 1:16 ResNet50 achieves 74.6% top-1 accuracy, which outperforms the previous state-of-the-art by 2.8%.

4.3. Comparison with Unstructured Sparsity

We also compare the performance of MaxQ with state-of-the-art unstructured sparsity methods using ResNet50. The results in Tab. 4 demonstrate that our MaxQ consistently achieves outstanding accuracy under various sparsity constraints. For example, ResNet50 with 1:4 and 2:8 sparse patterns achieve 77.3% and 77.2% top-1 accuracy respectively. Meanwhile, ResNet50 with 1:16 sparse pattern achieves 74.6% accuracy, surpassing STR and OptG over 4.2% and 2.1%. In contrast to unstructured sparsity, N:M organizes weights in a structured format, avoiding inefficient memory access and low computational density due to irregular distribution of weights. Additionally, the uniform distribution of N:M sparse weights ensures that computational loads are balanced when performing parallel computation and avoiding speed degradation due to load imbalance across threads. These exhibit the effectiveness and advantages of exploring N:M sparsity.

Model	Method	N:M	mAP
F-RCNN	Baseline	-	37.4
	SR-STE	2:4	38.2
	LBC	2:4	38.5
	MaxQ	2:4	**38.7**
	SR-STE	1:4	37.2
	LBC	1:4	37.3
	MaxQ	1:4	**37.7**

Table 5. Results for object detection on COCO benchmark.

Model	Method	N:M	Box mAP	Mask mAP
M-RCNN	Baseline	-	38.2	34.7
	SR-STE	2:4	39.0	35.3
	LBC	2:4	**39.3**	35.4
	MaxQ	2:4	39.2	**35.5**
	SR-STE	1:4	37.6	33.9
	LBC	1:4	37.8	34.0
	MaxQ	1:4	**38.3**	**34.4**

Table 6. Results for instance segmentation on COCO benchmark.

4.4. Object Detection and Instance Segmentation

In addition to the image classification, we conducted experiments on the challenging dataset COCO based on MMDetection [4], to exploit the generalization ability of MaxQ. For object detection, we employ classical models Faster R-CNN, and for instance segmentation, we use Mask R-CNN. Tab. 5 demonstrates that MaxQ consistently outperforms previous methods in object detection. For example, MaxQ yields a robust performance of 38.7 mAP at 2:4 sparse pattern, which exceeds the previous state-of-the-art LBC by 0.2 mAP and improves it's dense counterpart by 1.3 mAP. Similar trends are observed in instance segmentation, as shown in Tab. 6. Furthermore, MaxQ delivers results comparable to dense baseline models at a 1:4 structure sparsity. These results suggest that N:M sparse networks exhibit similar or better feature transfer capabilities than dense networks and can be effectively applied in downstream tasks.

4.5. Ablation Study

Method	Top-1
Baseline (SR-STE)	71.5%
+ Multi-Axis Query	74.2%
+ Incremental Pruning (**default**)	74.6%
+ Train 200 Epochs	75.2%

Table 7. Ablation study of different components in MaxQ.

We investigate the effectiveness of different components in the MaxQ through ablation studies. All the following

results are based on the ResNet50 model with 1:16 sparse pattern on the ImageNet dataset.

Components. In Tab. 7, we investigate the effectiveness of different components in MaxQ, namely multi-axis query and incremental pruning. The baseline is derived from SR-STE where the model is trained solely with the sparse-refined straight-through estimator. There is no weight importance identifying among blocks and incremental pruning stage. The multi-axis query identifies and enhances the weights with more importance using soft masks, preventing crucial weights from being overlooked. It ensures more effective updates for important weights during training, resulting in 2.7% accuracy improvement. When the percentage of N:M sparse block in each layer increases according to a specific schedule, instead of being fixed to 100% early in training, we can obtain a more optimal N:M sparse network. This dynamic approach further improves accuracy by 0.4%. Additionally, to achieve a fair comparison with ASP, we conduct an experiment to study the effect of training epochs. We train MaxQ for 200 epochs, the same ASP setup. The experiment result shows that longer training epochs lead to considerable performance gains, improving accuracy by 0.6% compared to the default settings. It's worth noting that for N:M sparse networks, the PPF training pipeline is unnecessary as it does not significantly improve its performance. On the contrary, applying soft and incremental strategies to train an N:M sparse network from scratch yields better results. Moreover, these strategies are more straightforward when compared to the PPF pipeline.

Model	N:M	t_i	t_f	$t_f - t_i$	Top-1	FLOPs(Train)
ResNet50	1:16	0	30	30	74.51%	0.74×(3.2e18)
	1:16	0	60	60	74.41%	0.78×
	1:16	0	90	90	74.55%	0.81×
	1:16	30	60	30	74.56%	0.81×
	1:16	30	90	60	74.52%	0.85×
	1:16	60	90	30	74.47%	0.89×

Table 8. Ablation study of different t_i and t_f in MaxQ.

Choice of t_i and t_f. We conduct ablation studies involving different t_i and t_f in MaxQ. To explain, larger t_i and $t_f - t_i$ indicate more weights can be sufficiently trained before applying N:M sparsity and a smoother sparsity process. And the total train FLOPs are positively correlated with t_i and t_f. As illustrated in Tab. 8, our method can achieve excellent performance across various t_i and t_f. Moreover, more training FLOPs does not bring a notable improvement to the final result. It suggests that the weight distribution characteristics of the trained dense network are not essential and explains why the PPF strategy for the ASP does not yield favorable results in the N:M sparsity networks. Superior N:M sparsity networks can be trained in one training pass by soft, dynamic and incremental strategies.

Sparsity strategy. We introduce two other strategies to investigate the efficiency of our strategy for incremental sparsity. t_i and t_f are set to 0 and 90 respectively. As shown in Fig. 5, both of inverse and N-M suffer severe performance degradation when the sparse ratio increases (when epoch increases to 90). Applying N:M sparsity first to weight blocks with larger ℓ_1-norm will bring more stable and efficient convergence than others. At the same time, our strategy achieves 74.55% top-1 accuracy on ImageNet, better than the inverse with 74.13% and the N-M with 74.18%.

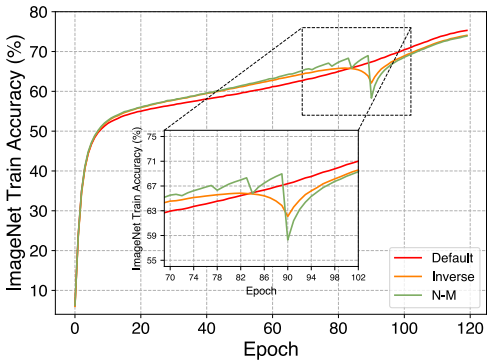

Figure 5. Convergence visualization for different strategies. Inverse means we firstly apply the N:M sparsity for blocks with smaller ℓ_1-norm. N-M means reducing $\left\|\mathbf{b}^t_{g,:}\right\|_0$ from M to N.

4.6. Performance Analysis

Model	N:M	Method	Train speed BS=128	Train speed BS=256	Top-1	FLOPs (Train)
ResNet50	-	Dense	798	884	77.3%	1×(3.2e18)
	2:4	SR-STE	**642**	**854**	77.0%	0.83×
		LBC	373	487	77.2%	**0.72**×
		MaxQ	507	732	**77.6%**	0.91×
	2:8	SR-STE	**625**	**862**	76.2%	0.74×
		LBC	382	512	76.5%	**0.53**×
		MaxQ	514	743	**77.2%**	0.86×
	1:16	SR-STE	**628**	**852**	71.5%	0.69×
		LBC	364	538	71.8%	**0.38**×
		MaxQ	502	725	**74.6%**	0.81×

Table 9. Train speed (NVIDIA RTX 3090, measured in samples/second/GPU), ImageNet top-1 accuracy and FLOPs (train) with different N:M sparse pattern and methods.

Model	N:M	Method	FP32	INT8	Acc Drop
ResNet50	2:4	MaxQ	77.6%	77.1%	0.5%
		SR-STE	77.1%	76.6%	0.5%

Table 10. Post-Training Quantization (PTQ) results on SR-STE and MaxQ for ResNet50 with 2:4 sparse pattern.

Inference. As mentioned earlier, MaxQ can be considered a self-structured re-parameterization process during inference. Thus, the soft masks and weights can be folded as constants and do not introduce any additional overhead during the inference. Furthermore, as the soft masks also adhere to the N:M sparse pattern, MaxQ does not disrupt the sparse pattern and can still leverage its benefits, resulting in favorable latency on Ampere GPUs.

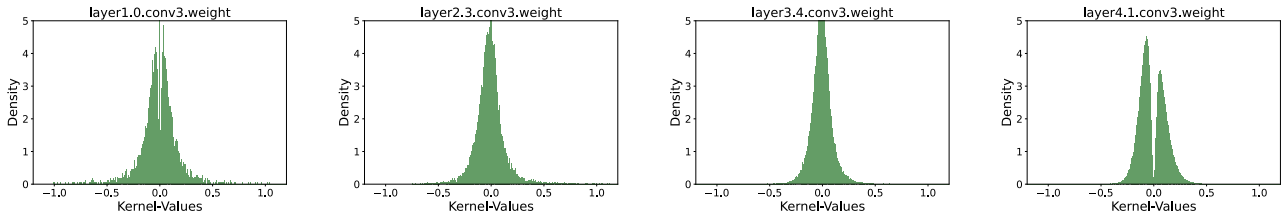

Figure 6. Parameter distribution from SR-STE at inference.

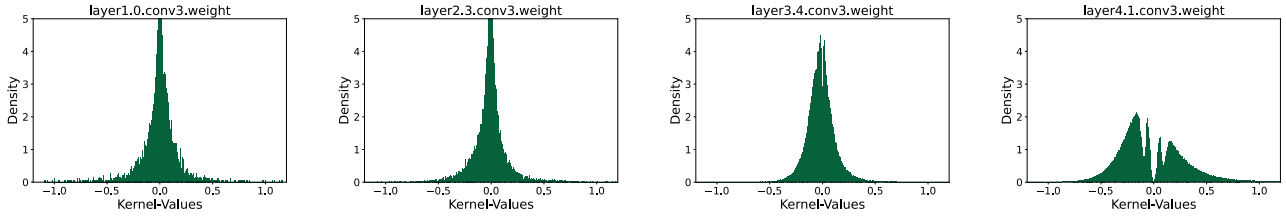

Figure 7. Parameter distribution from MaxQ at inference.

Training. The training efficiency of the algorithm is also an important aspect to consider. While many algorithms have smaller FLOPs theoretically, they may not be highly parallelized or have a large amount of memory access. Thus, there will be an increase but not a reduction in GPU days. To investigate the efficiency of various algorithms for N:M sparsity, we present the train speed, top-1 accuracy, and training FLOPs for SR-STE, LBC, and MaxQ in Tab. 9. For a fair comparison, their train speeds are tested with the same training script on the same machine with an NVIDIA RTX 3090 and automatic mixed precision. SR-STE is the fastest. MaxQ, although about 15% slower than SR-STE, achieves the highest top-1 accuracy among the three methods. The decrease in train speed primarily results from the multi-axis query, which incurs high memory access but involves minimal computational workload. Furthermore, although LBC has the theoretically least training FLOPs, it is the slowest among the three methods. Compared to the train speed of dense network, LBC is almost 50% slower for a batch size of 128 and 30% slower for a batch size of 256. In summary, MaxQ stands out as the most efficient algorithm, striking the best balance between training speed and accuracy.

4.7. Quantization

A structured re-parameterized network with simple PTQ may suffer from significant degradation in accuracy [6, 8], which is completely unusable. To gain insights into the characteristics of MaxQ quantization, we first visualize the weight distribution of SR-STE and MaxQ in Fig. 6 and Fig. 7 respectively. The weight distribution of SR-STE and MaxQ exhibits notable differences for the same model. Specifically, in shallower layers, MaxQ has a sharper weight distribution. While in deeper layers, it is smoother and has a broader dynamic range. We conducted quantization experiments on SR-STE and MaxQ for comparison. The results are presented in Tab. 10. To our surprise, our MaxQ demonstrates quantization-friendliness despite being perceived as a self-structured re-parameterized process. For ResNet50, MaxQ with a 2:4 sparse pattern can achieve 0.5% drop in the top-1 accuracy (77.6% $\rightarrow$ 77.1%), when it is quantized to INT8 using a simple uniform PTQ method provided by the Ascend Tensor Compiler (ATC). Similar results are observed for SR-STE. These experiments indicate that the weight distribution obtained by MaxQ are also applicable to quantize. Furthermore, the performance of the INT8 model can still be improved by the more advanced quantization methods, such as non-uniform PTQ and Quantization-Aware Traning (QAT).

5. Conclusion

N:M sparsity is a crucial method to reduce inference overhead and enable fast inference on NVIDIA Ampere GPUs. In this paper, we propose a novel Multi-Axis Query method, MaxQ, to identify the critical weights and build a high-performance N:M sparsity network. During the training, MaxQ employs a dynamic approach to generate soft N:M masks, which enhances the weights with more importance and ensures more effective updates for them. During the runtime, soft N:M masks can be folded into the network as constants, which will not cause any distortion to the sparse pattern or bring additional computational costs. Further, MaxQ follows a gradual sparse schedule by increasing the percentage of N:M weight blocks. It progressively allows the network to heal from sparsity, avoiding severe information loss and achieving stable and efficient convergence. Experiments on various vision tasks demonstrate that MaxQ can achieve notable performance gains over the state-of-the-art methods on multiple N:M sparse patterns and CNNs.

Acknowledgements This work is funded in part by the Key Research and Development Project of Zhejiang Province under Grant 2021C01035.

References

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

[2] Jihye Back, Namhyuk Ahn, and Jangho Kim. Magnitude attention-based dynamic pruning. *arXiv preprint arXiv:2306.05056*, 2023. 6

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 6

[5] Zhuangzhi Chen, Jingyang Xiang, Yao Lu, Qi Xuan, Zhen Wang, Guanrong Chen, and Xiaoniu Yang. Rgp: Neural network pruning through regular graph with edges swapping. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2023. 3

[6] Xiangxiang Chu, Liang Li, and Bo Zhang. Make repvgg greater again: A quantization-aware approach. *arXiv preprint arXiv:2212.01593*, 2022. 8

[7] Ioana Croitoru, Simion-Vlad Bogolin, and Marius Leordeanu. Unsupervised learning of foreground object segmentation. *International Journal of Computer Vision (IJCV)*, 127(9):1279–1302, 2019. 1

[8] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Kaiqi Huang, Jungong Han, and Guiguang Ding. Re-parameterizing your optimizers rather than architectures. In *The Eleventh International Conference on Learning Representations*, 2022. 8

[9] Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. Fast sparse convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14629–14638, 2020. 3

[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 (ICML)*, pages 2943–2952, 2020. 2, 3, 6

[11] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. 2

[12] Ross Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 1

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 580–587, 2014. 1

[14] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016. 2

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015. 1, 2, 3

[17] 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. 1, 5

[18] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 5

[19] 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. 2

[20] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2234–2240, 2018. 1

[21] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349, 2019. 1, 3

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

[23] Zejiang Hou, Minghai Qin, Fei Sun, Xiaolong Ma, Kun Yuan, Yi Xu, Yen-Kuang Chen, Rong Jin, Yuan Xie, and Sun-Yuan Kung. Chex: Channel exploration for cnn model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12287–12298, 2022. 3

[24] 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. 1, 5

[25] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems*, 34:21099–21111, 2021. 1

[26] Ryan Humble, Maying Shen, Jorge Albericio Latorre, Eric Darve, and Jose Alvarez. Soft masking for cost-constrained channel pruning. In *European Conference on Computer Vision*, pages 641–657. Springer, 2022. 3

[27] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learn-

able sparsity. In *International Conference on Machine Learning*, pages 5544–5555. PMLR, 2020. 2, 3, 6

[28] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019. 1, 3

[29] Bin Lin, Ningxin Zheng, Lei Wang, Shijie Cao, Lingxiao Ma, Quanlu Zhang, Yi Zhu, Ting Cao, Jilong Xue, Yuqing Yang, et al. Efficient gpu kernels for n: M-sparse weights in deep learning. *Proceedings of Machine Learning and Systems*, 5, 2023. 1

[30] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1529–1538, 2020. 1, 2, 3

[31] Mingbao Lin, Yuxin Zhang, Yuchao Li, Bohong Chen, Fei Chao, Mengdi Wang, Shen Li, Yonghong Tian, and Rongrong Ji. 1xn pattern for pruning convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 3

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014. 2, 5

[33] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1

[34] 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. 3

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

[36] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. 3

[37] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 1

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 5

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, pages 10347–10357, 2021. 1

[41] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016. 2

[42] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. *Advances in Neural Information Processing Systems*, 32, 2019. 6

[43] 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. 6

[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*, 2022. 1, 2, 5, 6

[45] Yuxin Zhang, Yiting Luo, Mingbao Lin, Yunshan Zhong, Jingjing Xie, Fei Chao, and Rongrong Ji. Bi-directional masks for efficient N: M sparse training. In *International Conference on Machine Learning*, pages 41488–41497. PMLR, 2023. 1

[46] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. In *International Conference on Learning Representations Workshop (ICLRW)*, 2017. 6