Bi-directional Masks for Efficient N:M Sparse Training

Yuxin Zhang*[1] Yiting Luo*[1] Mingbao Lin[2] Yunshan Zhong[1] Jingjing Xie[1] Fei Chao[1] Rongrong Ji[1 3 4]

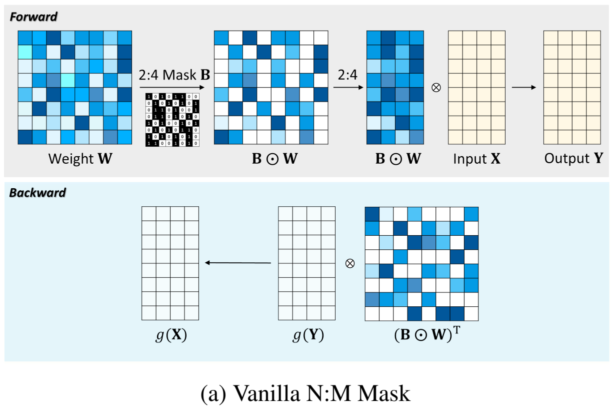
Abstract

We focus on addressing the dense backward propagation issue for training efficiency of N:M fine-grained sparsity that preserves at most N out of M consecutive weights and achieves practical speedups supported by the N:M sparse tensor core. Therefore, we present a novel method of Bi-directional Masks (Bi-Mask) with its two central innovations in: 1) Separate sparse masks in the two directions of forward and backward propagation to obtain training acceleration. It disentangles the forward and backward weight sparsity and overcomes the very dense gradient computation. 2) An efficient weight row permutation method to maintain performance. It picks up the permutation candidate with the most eligible N:M weight blocks in the backward to minimize the gradient gap between traditional uni-directional masks and our bi-directional masks. Compared with existing uni-directional scenario that applies a transposable mask and enables backward acceleration, our Bi-Mask is experimentally demonstrated to be more superior in performance. Also, our Bi-Mask performs on par with or even better than methods that fail to achieve backward acceleration. Project of this paper is available at `https://github.com/zyxxmu/Bi-Mask`.

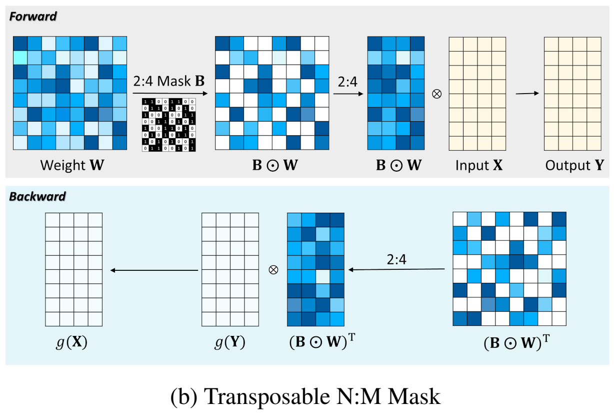
1. Introduction

The past decade has witnessed thriving deep neural networks (DNNs) in various machine learning applications (He et al., 2016; 2017a; Girshick et al., 2014). In large part, the prosperity is driven by increasing parameters and computations, which however, make DNN models too cumbersome to be deployed on resource-constrained edge devices such as cell phones and Internet-of-Things (IoT) devices. Therefore, the research community is sorely in need of technical renovation to compress the DNNs (Hubara et al., 2016; Tan & Le, 2019; Lin et al., 2020).

By removing redundant network weights (LeCun et al., 1989; Han et al., 2015; He et al., 2017b), network spar-



(a) Vanilla N:M Mask



(b) Transposable N:M Mask

Figure 1. Comparison between vanilla N:M mask and transposable N:M mask (2:4 case). The vanilla N:M mask (Zhou et al., 2021; Nvidia, 2020) generates sparse weights with N:M property in rows, leading to forward acceleration but remaining dense backward propagation as the weight transposition operation impairs N:M blocks. The transposable N:M mask (Hubara et al., 2021) generates sparse weights that have N:M property in both rows and columns, leading to forward & backward acceleration. Both methods consider only one sparse mask.

*Equal contribution [1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, Xiamen University, Xiamen, China [2]Tencent Youtu Lab, Shanghai, China [3]Institute of Artificial Intelligence, Xiamen University, Xiamen, China [4]Pengcheng Lab, Shenzhen, China. Correspondence to: Rongrong Ji <rrji@xmu.edu.cn>, Rongrong Ji <rrj@xmu.edu.cn>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

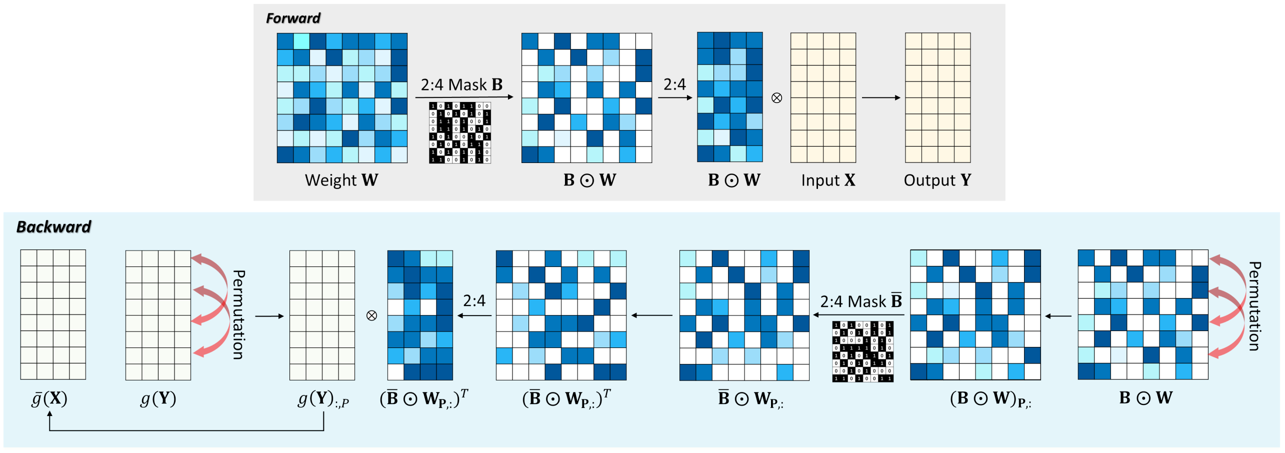

Forward
2:4 Mask $\mathbf{B}$
2:4
Weight $\mathbf{W}$
$\mathbf{B} \odot \mathbf{W}$
$\mathbf{B} \odot \mathbf{W}$
$\otimes$
Input $\mathbf{X}$
Output $\mathbf{Y}$
Backward
Permutation
2:4
2:4 Mask $\bar{\mathbf{B}}$
Permutation
$\bar{g}(\mathbf{X})$
$g(\mathbf{Y})$
$g(\mathbf{Y})_{:,P}$
$(\bar{\mathbf{B}} \odot \mathbf{W}_{\mathbf{P},:})^T$
$(\bar{\mathbf{B}} \odot \mathbf{W}_{\mathbf{P},:})^T$
$\bar{\mathbf{B}} \odot \mathbf{W}_{\mathbf{P},:}$
$(\mathbf{B} \odot \mathbf{W})_{\mathbf{P},:}$
$\mathbf{B} \odot \mathbf{W}$

Figure 2. Framework of the proposed Bi-direction Masks (Bi-Mask). It separately builds two N:M sparse masks in the forward and backward direction, thus enabling training acceleration in both directions. During backward propagation, Bi-Mask performs an efficient row permutation to make the sparse weights have more eligible N:M weight blocks before generating the backward mask.

sity has emerged as a piece of modern equipment to obtain a lightweight sparse model. Through removing individual weights at arbitrary positions, fine-grained sparsity is demonstrated to reach a high sparse ratio with performance guarantee (Han et al., 2015; Evci et al., 2020). Unfortunately, the resulting unstructured sparse weights hardly produce acceleration on off-the-shelf hardware. Coarse-grained sparsity is more hardware friendly as it typically removes an entire weight block (Ji et al., 2018; Meng et al., 2020) or convolution filter (Liu et al., 2019a; Lin et al., 2020). In comparison with fine-grained sparsity, the compressed model gains noticeable speedup, yet suffers more performance degradation. Therefore, it is a challenging yet valuable issue to simultaneously retain model performance of DNN models and achieve hardware acceleration.

Luckily, recent N:M fine-grained sparsity has provided a promising solution. By requiring at most N non-zero elements out of every M contiguous weights, N:M sparsity includes the performance advantage of fine-grained sparsity as well as practical acceleration thanks to the hardware innovation of N:M sparse tensor core (Ronny Krashinsky, 2020; Fang et al., 2022). Nvidia (Nvidia, 2020) has presented the ASP (APEX's Automatic Sparsity) paradigm that achieves 2:4 sparsity within three steps, unfolded as training a dense network, applying 2:4 fine-grained sparsity using magnitude-based pruning (Han et al., 2015), and re-training the sparse network. Despite the satisfying performance, ASP exhibits drawbacks in its tedious training cost as it contains dense network training and N:M sparse retraining. This largely prohibits the application of N:M sparsity when confronting with scarce training resources.

The above issue has been partially addressed by directly training an N:M sparse network from scratch (Zhou et al., 2021). Yet, the sparse tensor core is only utilized to accelerate the forward multiplication during training. As illustrated in Fig. 1a, the weight transposition operation in the backward impairs N:M blocks and thus fails to support acceleration in gradient calculation. To mitigate this, (Hubara et al., 2021) proposed N:M transposable mask, where a binary mask that indicates whether to remove weights is required to have N:M property along the rows and columns. Therefore, after performing transposition, it still satisfies the N:M format as shown in Fig. 1b. Unfortunately, the transposable requirement is observed to have more performance degradation, which is presumably caused by less flexibility of the sparsity pattern (Hubara et al., 2021). In Sec. 3.2, we further show severe performance degradation at a higher sparse level such as 1:8 and 1:16. We therefore reflect on this: how can we address the efficiency of N:M sparse training without a compromise on performance?

In this paper, we attempt to answer the above question by introducing a novel method of Bi-directional Masks (Bi-Mask) that performs surprisingly well without any N:M transposable constraint. Fig. 2 illustrates framework of our Bi-Mask. In particular, along the forward and backward directions, two separate binary masks are constructed according to the weight magnitude (Han et al., 2015). As a contrast, we require the forward mask to follow N:M property in its rows while in columns for the backward mask. By this way, we concurrently enable forward & backward acceleration from the N:M sparse tensor core. Also, the bi-directional masks benefit performance from more flexible sparsity pattern. Nevertheless, they also bring about deficiency of gradient gap since the backward mask modifies the gradient of forward loss. Given this issue, an efficient row permutation is further introduced before enforcing the backward mask. In detail, we first change row order of weight

Table 1. Advantage comparison between the vanilla N:M mask (Mask) (Nvidia, 2020; Zhou et al., 2021), the transposable N:M mask (T-Mask) (Hubara et al., 2021) and our proposed Bi-direction Mask (Bi-Mask) for N:M sparse training.

Advantage	Vanilla Mask	T-Mask	Bi-Mask
Forward Acceleration	✓	✓	✓
Backward Acceleration	✗	✓	✓
Performance Maintenance	✓	✗	✓

matrix and then pick up the permutation with the most eligible N:M weight blocks from a dozen of candidates. By changing column order of output gradient accordingly, we succeed in retaining the same outputs between with/without row permutation, and at the same time well reducing the gradient gap between uni-directional/bi-directional mask(s).

Our simple design of Bi-Mask turns out to achieve remarkable results. Besides forward & backward training acceleration, Bi-Mask well improves the performance of transposable mask (T-Mask) across different N:M patterns, benchmarks, and networks. For example, Bi-Mask achieves 71.5% Top-1 accuracy when training 1:16 sparse ResNet-50 on ImageNet, surpassing T-mask by 5.3%. More surprisingly, our approach achieves comparable or even better results than vanilla N:M methods, where the backward propagation can not be accelerated. For example, our Bi-Mask exceeds Top-1 accuracy of SR-STE (Zhou et al., 2021) by 0.4% when training 2:4 sparse ResNet-50 on ImageNet. Table 1 provides advantage comparison between different mask methods.

2. Related Work

2.1. Network Sparsity

Network sparsity has been one of the most effective tools to relieve the complexity of DNNs over the past decades (LeCun et al., 1989; Han et al., 2015). Pioneering works implement network sparsity in a fine-grained granularity where weights at arbitrary positions are removed to obtain a compact network. (Han et al., 2015) presented a classic three-step paradigm including pre-training a full network, removing low-magnitude weights, and fine-tuning the sparse networks. The lottery ticker hypothesis (Frankle & Carbin, 2019) further reveals the existence of randomly-initialized sparse networks that can be trained independently to compete with the performance of the dense model. In principle, the fine-grained network sparsity can maintain the performance of dense networks at an ultra-high sparse ratio like 90.0% (Mostafa & Wang, 2019; Blalock et al., 2020). Nevertheless, it receives very limited speedups since the resulting sparse networks are in unstructured formats, which barely take advantage of general hardware platforms.

Coarse-grained sparsity targets at removing entire weight blocks (Ji et al., 2018; Meng et al., 2020) or convolution filters (Liu et al., 2019b; Lin et al., 2020) to make the sparse networks compatible with off-the-shelf hardware. For instance, (Li et al., 2017) removed convolution filters with smaller ℓ_1 norm, while (Lin et al., 2020) considered the rank of feature maps as the filter importance measurement. Unfortunately, coarse-grained sparsity suffers severe performance drops at sparsity levels higher than 50% due to the flexibility constraint on network sparsity (Renda et al., 2021). Different from the existing sparsity granularity, this paper focuses on N:M fine-grained sparsity (Zhou et al., 2021; Sun et al., 2021; Pool & Yu, 2021), which preserves at most N out of M consecutive weights. In addition to performance maintenance, N:M sparsity is also able to obtain practical acceleration from the hardware innovation of N:M sparse tensor core (Nvidia, 2020; Fang et al., 2022).

2.2. Sparse Training

Sparse training serves as an effective tool to boost the performance of network sparsity (Hoefler et al., 2021; Evci et al., 2020; Sanh et al., 2020). It dynamically prunes and revives weights of the sparse networks during training according to specific criteria. For example, RigL (Evci et al., 2020) removes smaller-magnitude weights and revives weights with larger-magnitude gradients. Besides, sparse momentum (Dettmers & Zettlemoyer, 2019) considers magnitude of mean weight momentum as a guide to redistribute the sparse weights. In this paper, we focus on training N:M sparse networks. As a study mostly related to ours, the transposable N:M masks (Hubara et al., 2021) requires one single sparse mask with N:M blocks in both rows and columns such that the transposition in the backward also embraces hardware acceleration. In contrast, our method separately builds sparse masks in the forward and backward propagation without additional sparse constraints and gains significantly better performance under the same N:M case. Besides, (Pool & Yu, 2021) proposed to permute the input channel of pre-trained CNNs to maximally preserve the magnitude of N:M sparse networks. Very differently, our Bi-Mask permutes the row dimension of sparse weights that are trained from scratch, with a diverse object of obtaining more eligible N:M weight blocks to mitigate the gradient gap in the backward sparsity.

3. Methodology

3.1. Revisiting N:M Sparse Training

We first introduce some basic knowledge about the N:M fine-grained sparsity. Let $\mathbf{W} \in \mathbb{R}^{I \times J}$ be the parameter matrix from a specific network layer. Considering the input tensor $\mathbf{X}$, the forward propagation represented with form of matrix multiplication can be formulated as:

$$\mathbf{Y} = \mathbf{W} * \mathbf{X}, \tag{1}$$

where $\mathbf{Y}$ is the output tensor and $*$ is the matrix multiplication operation. N:M sparsity forces at most N out of M consecutive weights in the rows of $\mathbf{W}$ to have non-zero values. The sparsity can be achieved via a binary matrix $\mathbf{B} \in \{0, 1\}^{I \times J}$ where a block of every M contiguous elements contains at most N as:

$$\|\mathbf{B}_{i,j:j+\mathrm{M}}\|_0 \leq \mathrm{N}, \tag{2}$$

in which $i = 1, 2, 3, ..., I$ and $j = 1, \mathrm{M}, 2\mathrm{M}, ..., J$. Then, the sparse forward propagation can be formulated as:

$$\mathbf{Y} = (\mathbf{B} \odot \mathbf{W}) * \mathbf{X}, \tag{3}$$

where $\odot$ denotes the element-wise multiplication. Since $\mathbf{B} \odot \mathbf{W}$ meets N:M requirement, the matrix multiplication with $\mathbf{X}$ can be efficiently implemented by the N:M sparse tensor core, as illustrated in Fig. 1a.

N:M sparse training starts from randomly-initialized networks (Zhou et al., 2021; Zhang et al., 2022), thus avoiding heavy burden of pre-training a dense model (Nvidia, 2020). We base our study on the popular SR-STE (Zhou et al., 2021) for N:M sparse training, simply illustrated for ease of understanding in the following. During forward propagation, it adapts the binary mask $\mathbf{B}$ at each iteration as:

$$\mathbf{B}_{i,j+m} = \begin{cases} 0, \text{ if } |\mathbf{W}_{i,j+m}| < \mathrm{Top}(|\mathbf{W}_{i,j:j+\mathrm{M}}|, \mathrm{N}), \\ 1, \text{ otherwise}, \end{cases} \tag{4}$$

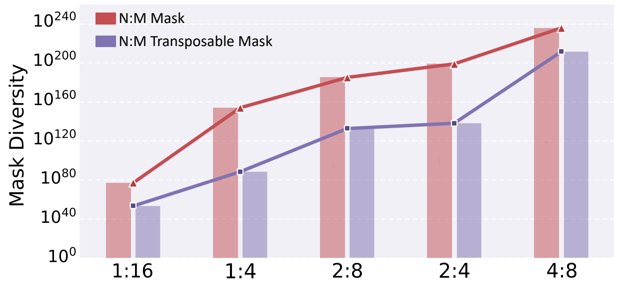
where $1 \leq m \leq \mathrm{M}$, $|\cdot|$ represents the absolute function, and $\mathrm{Top}(|\mathbf{W}_{i,j:j+\mathrm{M}}|, \mathrm{N})$ returns the N-th largest value within $|\mathbf{W}_{i,j:j+\mathrm{M}}|$. Therefore, we obtain the forward binary mask according to the weight magnitude in each block. During backward propagation, the gradients of $\mathbf{B} \odot \mathbf{W}$ are directly passed to $\mathbf{W}$ according to the straight-through-estimator (STE) (Bengio et al., 2013).

3.2. Rethinking the Transposable N:M Mask

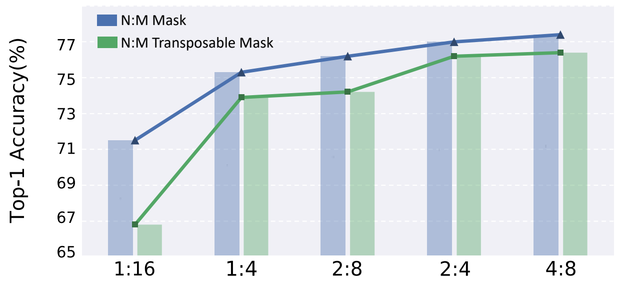
The above sparse mask is indeed uni-directional towards forward propagation. By forming N:M blocks in rows of the mask, Eq. (3) permits forward acceleration from the N:M sparse tensor core between the weights and inputs. Unfortunately, such a vanilla mask crashes backward acceleration due simply to the transposition operation. To explain, the gradient in the backward propagation is computed as:

$$g(\mathbf{X}) = (\mathbf{B} \odot \mathbf{W})^T * g(\mathbf{Y}), \tag{5}$$

where $g(\cdot)$ denotes the gradient with respect to its input. The above equation requires $(\mathbf{B} \odot \mathbf{W})^T$ to have N:M blocks in rows for accelerating multiplication with $g(\mathbf{Y})$, however, it is in columns on account of the transposition operation. Thus, the backward propagation remains dense and fails to be accelerated, as illustrated in Fig. 1a.



(a) Flexibility Comparison



(b) Performance Comparison

Figure 3. Comparison between vanilla mask and transposable mask including (a) flexibility measured by mask diversity (Hubara et al., 2021) and (b) performance of training sparse ResNet-50 (He et al., 2016) on ImageNet (Deng et al., 2009).

To address this issue, (Hubara et al., 2021) presented transposable N:M mask that is required to satisfy row-wise and column-wise N:M blocks such that the transposition also undertakes an important mission of N:M property in rows. Consequently, the binary mask $\mathbf{B}$ is constrained as:

$$\|\mathbf{B}_{i,j:j+\mathrm{M}}\|_0 \leq \mathrm{N}, \quad \|\mathbf{B}_{k:k+\mathrm{M},l}\|_0 \leq \mathrm{N}, \tag{6}$$

where $i = 1, 2, 3, ..., I$, $j = 1, \mathrm{M}, 2\mathrm{M}, ..., J$, $k = 1, M, 2M, ..., I$, and $l = 1, 2, 3, ..., J$. Besides, (Hubara et al., 2021) further introduced a 2-approximation algorithm to reduce complexity of finding the transposable mask.

Here we rethink the transposable pursuit for N:M sparse training. Although it enables backward acceleration, the flexibility of sparse networks is greatly restricted, which comes at the cost of performance degradation. We first report the flexibility comparison between vanilla mask and transposible mask under different N:M cases. Fig. 3a measures the flexibility using mask diversity that calculates the number of all possible masks under a given N:M mask case (Hubara et al., 2021). We can see a drastic flexibility degradation, in particular in cases of a small N or M. As a consensus (Gale et al., 2019; Nvidia, 2020), more restrictions on sparse patterns incur worse performance of sparse networks. For example, unstructured sparsity (Han et al., 2015) that removes arbitrary weights generally performs much better than structured sparsity (Li et al., 2017) that removes entire filter weights. Consequently, severe performance occurs in transposable mask in comparison with the vanilla method, as we experimentally verify in Fig. 3b, notably very poor 1:8 and 1:16. The uni-directional masks, either vanilla or transposable, do not accomplish N:M backward acceleration without a compromise on performance. Therefore, in what follows, we address this issue from the perspective of bi-directional masks.

3.3. Bi-directional N:M Masks

In this section, we formally present our Bi-directional N:M masks (Bi-Mask). As its name suggests, our Bi-Mask disentangles the forward & backward weight sparsity by involving two different masks during N:M sparse training. Concretely speaking, in the forward direction, we count on the vanilla N:M mask $\mathbf{B}$ from Eq. (2) that calls for N:M in rows to ensure the forward acceleration and results in better flexibility than the transposable N:M mask as we report in Fig. 3a. Very differently, we additionally build another mask $\bar{\mathbf{B}} \in \{0,1\}^{I \times J}$ in the backward direction with the N:M requirement on its columns as:

$$\|\bar{\mathbf{B}}_{k:k+\mathrm{M},l}\|_0 \leq \mathrm{N}, \tag{7}$$

in which $k = 1, \mathrm{M}, 2\mathrm{M}, ..., I$, and $l = 1, 2, 3, ..., J$. In this fashion, the backward acceleration is supported as well without a compromise on the flexibility of backward mask, and the backward gradient $g(\mathbf{X})$ in Eq. (5) is represented by the following approximation:

$$\bar{g}(\mathbf{X}) = (\bar{\mathbf{B}} \odot \mathbf{W})^T * g(\mathbf{Y}). \tag{8}$$

Nevertheless, the forward $\mathbf{B}$ requires gradient of $g(\mathbf{X})$ for our Bi-Mask, which yields a gradient gap between practical bi-directional gradient $\bar{g}(\mathbf{X})$ and ideal uni-directional gradient $g(\mathbf{X})$. To solve this issue, we adapt the backward mask $\bar{\mathbf{B}}$ to the magnitudes of masked weights during sparse training as follows:

$$\bar{\mathbf{B}}_{k+m,l} = \begin{cases} 0, \text{ if } |(\mathbf{B} \odot \mathbf{W})_{k+m,l}| \\ \qquad < \mathrm{Top}(|(\mathbf{B} \odot \mathbf{W})_{k:k+\mathrm{M},l}|, \mathrm{N}), \\ \mathbf{B}_{k+m,l}, \text{ otherwise}, \end{cases} \tag{9}$$

where $k = 1, \mathrm{M}, 2\mathrm{M}, ..., I$, $l = 1, 2, ..., J$, and $1 \leq k \leq \mathrm{M}$. For a deeper analysis, it is easy to understand that $\mathbf{B}_{k+m,l} = 0$ is a fully not necessary condition of $\mathbf{B}_{k+m,l} = 0$. That is, the event $\mathbf{B}_{k+m,l} = 0$ will produce the event $\mathbf{B}_{k+m,l} = 0$, but is not the only way for $\mathbf{B}_{k+m,l} = 0$ to occur.

The rationale behind Eq. (9) is two-fold: 1) It maximizes the similarity of forward and backward masks by setting $\bar{\mathbf{B}}_{k+m,l} = \mathbf{B}_{k+m,l}$ if the magnitude of $\mathbf{W}_{k+m,l}$ is beyond the top-N largest. 2) Applying our backward mask does not affect the updating of these weights with zero forward masks since $\mathbf{B}_{k+m,l} = 0$ always results in $\bar{\mathbf{B}}_{k+m,l} = 0$.

Algorithm 1 Bi-Mask for Efficient N:M Sparse Training.

```
Require : Iteration interval $\Delta T$, permutation candidate number $K$, weight matrix $\mathbf{W}$, training iteration $T$.
Output : Trained sparse weights $\mathbf{W} \odot \mathbf{B}$.
1  for $t \in [1, 2, \dots, T]$ do
2      Obtain the forward mask $\mathbf{B}$ via Eq. (4);
3      Forward propagation via Eq. (3);
4      if $t \% \Delta T = 0$ then
5          Randomly generate $K$ permutations and pick up the one as $P$ with the most eligible N:M blocks;
6      end
7      Obtain the backward mask $\bar{\mathbf{B}}$ via Eq. (11);
8      Backward propagation via Eq. (10);
9      Update via the SGD optimizer;
10 end
11 Return $\mathbf{W} \odot \mathbf{B}$.
```

Unfortunately, it is a possibility that $\bar{\mathbf{B}}_{k+m,l} = 0$ does not necessarily result from $\mathbf{B}_{k+m,l} = 0$, in which case gradients of some non-zero masked weights are mistakenly eliminated, incurring performance degradation.

To decrease this possibility, we continue a row permutation method along the row dimension of $\mathbf{B} \odot \mathbf{W}$. Our major motivations are also two-fold: 1) We can see from Eq. (9) that the resulting mask block $\bar{\mathbf{B}}_{k:k+\mathrm{M},l}$ would exactly match with $\mathbf{B}_{k:k+\mathrm{M},l}$ if $(\mathbf{B} \odot \mathbf{W})_{k:k+\mathrm{M},l}$ has N:M sparsity, and no gradient gap would occur. 2) Performing row permutation of $\mathbf{B} \odot \mathbf{W}$ improves the number of eligible N:M blocks as illustrated in Fig. 2. Importantly, it does not violate the gradient computation. Denoting $P \in \mathbb{N}^I$ as a permutation of $\{1, 2, 3, ..., I\}$, the backward gradient $\bar{g}(\mathbf{X})$ in Eq. (8) can be equally computed as:

$$\bar{g}(\mathbf{X}) = \big(\bar{\mathbf{B}} \odot (\mathbf{W}_{P,:})\big)^T * \big(g(\mathbf{Y})_{:,P}\big), \tag{10}$$

where the backward mask $\bar{\mathbf{B}}$ is computed based on the permutated $(\mathbf{B} \odot \mathbf{W})_{P,:}$ accordingly:

$$\bar{\mathbf{B}}_{k+m,l} = \begin{cases} 0, \text{ if } \Big|\big((\mathbf{B} \odot \mathbf{W})_{P,:}\big)_{k+m,l}\Big| \\ \qquad < \mathrm{Top}\Big(\Big|\big((\mathbf{B} \odot \mathbf{W})_{P,:}\big)_{k:k+\mathrm{M},l}\Big|, \mathrm{N}\Big), \\ (\mathbf{B}_{P,:})_{k+m,l}, \text{ otherwise}. \end{cases} \tag{11}$$

Therefore, we only need to find a permutation P that results in more eligible N:M blocks in each column of $(\mathbf{B} \odot \mathbf{W})_{P,:}$. More N:M blocks decrease the possibility of eliminating gradients of non-zero masked weights. To avoid the cumbersome $I!$ possible permutations at each training iteration, we update a good permutation at a regularly spaced interval of every ΔT training iterations, and at each interval pick up the one that leads to the most eligible N:M blocks from randomly generating K permutation candidates.

In Sec. 4.4, we analyze that the permutation candidate number $K = 100$ already returns good performance. Compared with the aforementioned 2-approximation algorithm for the transposable N:M mask (Hubara et al., 2021), our method brings negligible runtime burden as we experimentally reported in Sec. 4.3. Our algorithm presented in this paper is outlined in Alg. 1.

4. Experimentation

4.1. Settings

Datasets and Backbones. We conduct experiments on representative benchmarks for image classification. For small-scale dataset, we choose the CIFAR-10 dataset (Krizhevsky et al., 2009), which contains 60,000 32×32 color images from 10 different classes, with 6,000 images for each class. For large-scale dataset, we choose the challenging ImageNet (Deng et al., 2009), which contains over 1.2 million images for training and 50,000 validation images in 1,000 categories. On CIFAR-10, we train N:M sparse ResNet-32 (He et al., 2016) and MobileNet-V2 (Sandler et al., 2018). On ImageNet, we train N:M sparse ResNet-18/50 (He et al., 2016) and DeiT-small (Touvron et al., 2021). We compare our Bi-Mask with classic N:M sparse training methods including ASP (Nvidia, 2020) and SR-STE (Zhou et al., 2021) that fail backward acceleration, and transposable N:M mask (T-Mask) (Hubara et al., 2021) that has backward acceleration. Top-1 classification accuracy is reported for comparison on both datasets.

Implementation Details. Our implementation of Bi-Mask is based on the PyTorch framework (Paszke et al., 2019). All experiments are conducted on the NVIDIA Tesla A100 GPUs. The training iteration interval ΔT is set to 100 and the number of permutation candidates K is set to 100. We use the stochastic gradient descent (SGD) optimizer to perform sparse training. In the first 5 training epochs, the learning rate linearly increases from 0 to 0.1 and then is decayed using the cosine annealing (Loshchilov & Hutter, 2017). The momentum and batch size are respectively set to 0.9 and 256. On CIFAR-10, we train all networks for 300 epochs with a weight decay of 1×10^{-3}. On ImageNet, we follow (Zhou et al., 2021) to train ResNet-18/50 for a total of 120 epochs. For DeiT-small, we follow (Zhang et al., 2022) to train for 300 epochs in total using the timm framework (Wightman, 2019).

4.2. Comparison on CIFAR-10

ResNet-32. We first apply our Bi-Mask to train ResNet-32 model. The quantitative results are reported in Table 2. We can see from the table that the proposed Bi-Mask yields significantly better performance than the transposable mask at all N:M cases, and achieves comparable performance with the vanilla N:M mask that fails to achieve backward acceleration. For example, Bi-Mask obtains 94.78% Top-1 accuracy at 2:4 sparse pattern, surpassing SR-STE and T-Mask by 0.10% and 0.26%. Therefore, these accuracy results well demonstrate the efficacy of our Bi-Mask.

Table 2. Comparison between different methods for training the N:M sparse ResNet-32 on CIFAR-10.

Method	N:M Pattern	Top-1 Accuracy (%)	Forward Acceleration	Backward Acceleration
Baseline	-	94.52	✗	✗
SR-STE	2:4	94.68	✓	✗
T-Mask	2:4	94.52	✓	✓
Bi-Mask	2:4	**94.78**	✓	✓
SR-STE	1:4	**94.52**	✓	✗
T-Mask	1:4	94.04	✓	✓
Bi-Mask	1:4	94.43	✓	✓
SR-STE	1:16	**92.92**	✓	✗
T-Mask	1:16	92.02	✓	✓
Bi-Mask	1:16	92.77	✓	✓

Table 3. Comparison between different methods for training the N:M sparse MobileNet-V2 on CIFAR-10.

Method	N:M Pattern	Top-1 Accuracy (%)	Forward Acceleration	Backward Acceleration
Baseline	-	94.43	✗	✗
SR-STE	2:4	94.26	✓	✗
T-Mask	2:4	94.12	✓	✓
Bi-Mask	2:4	**94.46**	✓	✓
SR-STE	1:4	**94.48**	✓	✗
T-Mask	1:4	93.81	✓	✓
Bi-Mask	1:4	94.28	✓	✓
SR-STE	1:16	**93.14**	✓	✗
T-Mask	1:16	90.12	✓	✓
Bi-Mask	1:16	92.48	✓	✓

MobileNet-V2. We further investigate the effectiveness of Bi-Mask for training N:M sparse MobileNet-V2, a prevailing network with a compact design of depth-wise separable convolution. Table 3 again suggests a significantly higher accuracy of Bi-Mask compared with T-Mask under the same backward acceleration. For instance, Bi-Mask maintains the performance of the dense model at 1:4 pattern, while T-Mask suffers apparent accuracy drops (94.28%, 94.43%, and 93.81%) for Bi-Mask, dense model, and T-Mask, respectively).

4.3. Comparison on ImageNet

ResNet-18. Table 4 shows the performance comparison of different methods for training 2:4 sparse ResNet-18 on ImageNet. Compared with T-Mask (Hubara et al., 2021), the proposed Bi-Mask achieves 1.6% performance gains.

Table 4. Comparison between different methods for training the N:M sparse ResNet-18 on ImageNet.

Method	N:M Pattern	Top-1 Accuracy (%)	Forward Acceleration	Backward Acceleration
Baseline	-	70.9	✗	✗
ASP	2:4	69.9	✓	✗
SR-STE	2:4	71.2	✓	✗
T-Mask	2:4	69.2	✓	✓
Bi-Mask	2:4	**70.8**	✓	✓

Table 5. Comparison between different methods for training the N:M sparse ResNet-50 on ImageNet.

Method	N:M Pattern	Top-1 Accuracy (%)	Forward Acceleration	Backward Acceleration
Baseline	-	77.1	✗	✗
ASP	2:4	76.8	✓	✗
SR-STE	2:4	77.0	✓	✗
T-Mask	2:4	76.2	✓	✓
Bi-Mask	2:4	**77.4**	✓	✓
Baseline	-	77.1	✗	✗
SR-STE	4:8	77.4	✓	✗
T-Mask	4:8	77.1	✓	✓
Bi-Mask	4:8	**77.5**	✓	✓
Baseline	-	77.1	✗	✗
SR-STE	1:4	75.3	✓	✗
T-Mask	1:4	73.8	✓	✓
Bi-Mask	1:4	**75.6**	✓	✓
Baseline	-	77.1	✗	✗
SR-STE	2:8	76.2	✓	✗
T-Mask	2:8	73.6	✓	✓
Bi-Mask	2:8	**76.3**	✓	✓
Baseline	-	77.1	✗	✗
SR-STE	1:16	71.5	✓	✗
T-Mask	1:16	66.4	✓	✓
Bi-Mask	1:16	**71.5**	✓	✓

Notably, Bi-Mask even surpasses ASP (Nvidia, 2020) by 0.9%, later of which fails backward acceleration. Therefore, the superiority of our proposed Bi-Mask on the large-scale dataset is validated.

ResNet-50. We further show the performance of training N:M sparse ResNet-50 on ImageNet. As shown in Table 5, Bi-Mask beats all the competitors across all N:M cases with the same or superior acceleration results. In particular, in comparison with SR-STE that gets stuck in dense backward propagation, Bi-Mask results in backward acceleration, meanwhile shows the best performance. For example, it surpasses SR-STE by 0.3% at 1:4. As for T-Mask that also accelerates the backward propagation, our T-Mask shows superior performance in particular to the cases with a smaller N value. As analyzed in Sec. 3.2, a small N or M greatly degrades the mask flexibility of T-Mask, therefore severe performance drops occur.

Table 6. Time comparison (s) between T-Mask and Bi-Mask for searching N:M masks of ResNet-50 at different patterns.

Method	2:4	4:8	1:4	2:8	1:16
T-Mask	155.1	193.2	168.6	200.1	278.4
Bi-Mask	15.5	14.3	13.6	15.7	15.0

Table 7. Comparison between different methods for training the N:M sparse DeiT-small on ImageNet.

Method	N:M Pattern	Top-1 Accuracy (%)	Forward Acceleration	Backward Acceleration
Baseline	-	79.8	✗	✗
SR-STE	2:4	75.7	✓	✗
T-Mask	2:4	71.5	✓	✓
Bi-Mask	2:4	**77.6**	✓	✓

Next, we compare the efficiency between searching the optimal N:M transposable mask (Hubara et al., 2021) and our permutation for the backward mask. We report the runtime for searching ResNet-50 with different N:M cases on one NVIDIA Tesla A100 GPU. Table 6 suggests superior efficiency of our Bi-Mask. For example, it takes negligible 15.0s for our Bi-Mask to find a good permutation at 1:16. As a contrast, T-Mask requires around 278.4s under the same N:M setting. Given the efficiency and accuracy, the efficacy of Bi-Mask is evident.

DeiT-small. We further continue to compare the efficacy of Bi-Mask for training 2:4 sparse DeiT, an advanced vision transformer model. As manifested in Table 7, the proposed Bi-Mask consistently obtains the best Top-1 under different N:M cases, with its additional merit in backward acceleration for N:M sparse training. For instance, Bi-Mask improves the Top-1 accuracy of SR-STE by 1.7% at 2:4, and gains 5.9% Top-1 improvements over off-the-shelf T-Mask. The above observations well demonstrate the ability of Bi-Mask in compressing and accelerating the recent advanced vision transformer models.

4.4. Performance Study

In this section, we investigate the performance of Bi-Mask to respectively explore the effectiveness of its components. All the experimental results are conducted on ImageNet using ResNet-18.

Permutation Updating. We first perform ablation studies of the permutation updating. In Fig. 4, we examine the performance of Bi-Mask under different training iteration intervals $\Delta T \in [1, 10, 100, 1000]$ and permutation candidate number $K \in [10, 100, 1000]$. As can be observed, the best accuracy is obtained when the permutation updating is performed every 100 training iterations. To analyze, small intervals incurs unstable sparse training as the typology

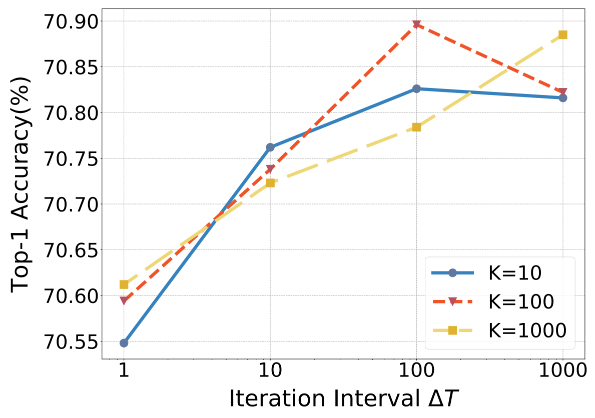


Figure 4. Performance influence of the training iteration inverval ΔT and candidation number K in our permutation updating.

of computing graph change in an excessive frequency. In contrast, large intervals outdate the optimal permutation, thus also leading to worse performance. Besides, the performance becomes saturated when the candidate number reaches 100 or more. The result shows that it is unnecessary to construct a time-consuming greedy algorithm to find out the optimal permutation.

Binarization Criteria. We further investigate the binarization criteria used to force the transposed weights to be N:M sparse after the permutation. Recall we opt magnitudes of sparse weights in Eq. (11) for a better fit with the forward mask to reduce the gradient gap. For comparison, we consider three variants including 1) preserving weights with the Top-N largest magnitudes of their gradients; 2) sampling N weights from the multinomial probability distribution according to their magnitudes; 3) randomly preserving N weights. Table 8 manifests our experimental results for the comparison. We can observe that the variants reduce the accuracy more or less. Among them, random one incurs the most performance drops since it severely enlarges the dissimilarity between the forward and backward masks and causes great gradient gaps. As for our weight magnitude, it well complies with the forward mask settings, therefore a better result can be observed.

Ablation Study. To further understand the effect of each component in our Bi-Mask, we conduct an ablation study by starting from our training baseline of the vanilla N:M mask (Zhou et al., 2021) (denoted by Baseline), and gradually including the backward mask and permutation updating. Table 9 shows that our backward mask enables backward acceleration with a Top-1 accuracy drop of 0.3%. As an analysis, the little degradation mainly yields from that our backward mask sometimes mistakenly eliminate gradients of some non-zero masked weights, as discussed in Sec. 3.3. After further adding our proposed permutation updating, the performance of sparse ResNet-18 even increases by 0.3% on the basis of the Baseline method. This is because our

Table 8. Performance of different binarization criteria for backward mask in Bi-Mask.

Model	Criteria	Pattern	Top-1 Accuracy(%)
ResNet-18	Gradient Magnitude	2:4	70.56
ResNet-18	Multinomial Sampling	2:4	70.42
ResNet-18	Random	2:4	67.76
ResNet-18	Weight Magnitude	2:4	70.73

Table 9. Ablation study for the proposed Bi-Mask.

Model	Criteria	Pattern	Top-1 Accuracy(%)
ResNet-18	Baseline	2:4	70.5
ResNet-18	+ Backward Mask	2:4	70.2
ResNet-18	+ Permutation Updating	2:4	70.8

permutation updating operation results in more eligible N:M blocks and reduce the possibility of incorrect gradient elimination. In conclusion, each part of our proposed Bi-Mask in this paper plays a unique role in boosting the performance of our N:M sparse training.

5. Limitations

In this section, we further discuss unexplored limitations of Bi-Mask in this paper, which will be our major future focuses. First, following the compared methods, we only train N:M sparse networks on the image classification task. More efforts can be made to verify the effectiveness of Bi-Mask on other tasks such as object detection. Besides, we only explore the acceleration on the forward and backward propagation, while accelerating the update phase of weights (Chmiel et al., 2022) remains to be excavated in the near future. At last, our random generation for the permutation does not always guarantee to maximize the number of N:M blocks. Though it does not damage the overall performance, a better solution is expected to be explored for possibly locating the best permutation.

6. Conclusion

In this work, we have presented a novel Bi-direction Mask (Bi-Mask) for efficient N:M find-grained sparse neural network training. Instead of imposing a transposable constraint on the N:M sparse mask, our Bi-Mask independently builds masks in the forward and backward directions. The mask in the backward direction is obtained through an efficient permutation in the weight rows and a following magnitude-based pruning to enable acceleration on the N:M sparse tensor core. Extensive experiments have demonstrated the superiority of Bi-Mask over several SOTAs. Particularly, Bi-Mask surpasses its competitor by a large margin under the same acceleration effects, and can even perform on par or even better than off-the-shelf methods that often fail to achieve backward acceleration.

Acknowledgement

This work was supported by National Key R&D Program of China (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

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

Chmiel, B., Hubara, I., Banner, R., and Soudry, D. Optimal fine-grained n: M sparsity for activations and neural gradients. *arXiv preprint arXiv:2203.10991*, 2022.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Dettmers, T. and Zettlemoyer, L. Sparse networks from scratch: Faster training without losing performance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning (ICML)*, pp. 2943–2952, 2020.

Fang, C., Zhou, A., and Wang, Z. An algorithm–hardware co-optimized framework for accelerating n: M sparse transformers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(11):1573–1586, 2022.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 580–587, 2014.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1135–1143, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. Mask r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, 2017a.

He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1389–1397, 2017b.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22:1–124, 2021.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

Hubara, I., Chmiel, B., Island, M., Banner, R., Naor, J., and Soudry, D. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Ji, Y., Liang, L., Deng, L., Zhang, Y., Zhang, Y., and Xie, Y. Tetris: Tile-matching the tremendous irregular sparsity. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

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

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 598–605, 1989.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017.

Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., and Shao, L. Hrank: Filter pruning using high-rank feature map. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1529–1538, 2020.

Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, T. K.-T., and Sun, J. Metapruning: Meta learning for automatic neural network channel pruning. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 3296–3305, 2019a.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *International Conference on Learning Representations (ICLR)*, 2019b.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

Meng, F., Cheng, H., Li, K., Luo, H., Guo, X., Lu, G., and Sun, X. Pruning filter in filter. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:17629–17640, 2020.

Mostafa, H. and Wang, X. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning (ICML)*, pp. 4646–4655, 2019.

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8026–8037, 2019.

Pool, J. and Yu, C. Channel permutations for n: M sparsity. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Renda, A., Frankle, J., and Carbin, M. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations (ICLR)*, 2021.

Ronny Krashinsky, O. G. e. a. Nvidia ampere sparse tensor core. https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/, 2020.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

Sanh, V., Wolf, T., and Rush, A. M. Movement pruning: Adaptive sparsity by fine-tuning. *arXiv preprint arXiv:2005.07683*, 2020.

Sun, W., Zhou, A., Stuijk, S., Wijnhoven, R., Nelson, A. O., Corporaal, H., et al. Dominosearch: Find layer-wise fine-grained n: M sparse schemes from dense neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pp. 6105–6114. PMLR, 2019.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, pp. 10347–10357, 2021.

Wightman, R. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Zhang, Y., Lin, M., Lin, Z., Luo, Y., Li, K., Chao, F., Wu, Y., and Ji, R. Learning best combination for efficient n: M sparsity. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhou, A., Ma, Y., Zhu, J., Liu, J., Zhang, Z., Yuan, K., Sun, W., and Li, H. Learning n: M fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations (ICLR)*, 2021.