Advancing Dynamic Sparse Training by Exploring Optimization Opportunities

Jie Ji [*1] **Gen Li** [*1] **Lu Yin** [2] **Minghai Qin** [1] **Geng Yuan** [3] **Linke Guo** [1] **Shiwei Liu** [4] **Xiaolong Ma** [1]

Abstract

Dynamic Sparse Training (DST) has been effectively addressing the substantial training resource requirements of increasingly large Deep Neural Networks (DNNs). Characterized by its dynamic "train-prune-grow" schedule during training, DST implicitly develops a bi-level structure for training the weights while discovering a subnetwork topology. However, such a structure is consistently overlooked by the current DST algorithms for further optimization opportunities, and these algorithms, on the other hand, solely optimize the weights while determining masks heuristically. In this paper, we extensively study DST algorithms and argue that the training scheme of DST naturally forms a bi-level problem in which the updating of weight and mask is interdependent. Based on this observation, we introduce a novel efficient training framework called BiDST, which for the first time, introduces bi-level optimization methodology into dynamic sparse training domain. Unlike traditional partial-heuristic DST schemes, which suffer from sub-optimal search efficiency for masks and miss the opportunity to fully explore the topological space of neural networks, BiDST excels at discovering excellent sparse patterns by optimizing mask and weight simultaneously, resulting in maximum 2.62% higher accuracy, $2.1\times$ faster execution speed, and $25\times$ reduced overhead. Code available at https://github.com/jjsrf/BiDST-ICML2024.

1. Introduction

Deep neural networks (DNNs) have transformed traditional machine learning technology into an intelligent and scalable ecosystem, fueling a wide range of challenging tasks (Ren et al., 2015; Li et al., 2023a; Yin et al., 2023; Kirillov et al., 2023). The arrival of the Artificial General Intelligence (AGI) era coupled with extensive data has catalyzed the development of larger model sizes and an exponential increase in computational requirements, resulting in a great challenge for end-users when implementing edge intelligence locally.

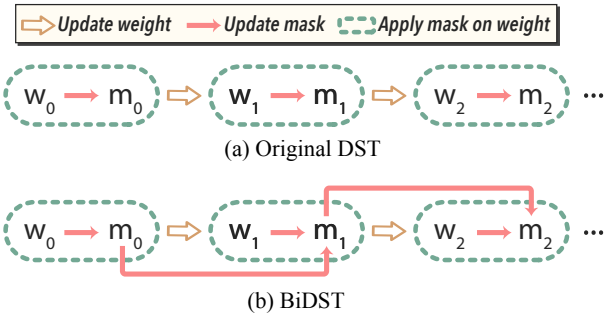


Figure 1: Training and mask update comparison between (a) original DST methods, and (b) the proposed BiDST.

A straightforward yet effective solution is to introduce *sparsity* into DNN training (Deng et al., 2020). As pointed out in Bellec et al. (2018) and Mocanu et al. (2018), weight sparsity can be leveraged to potentially satisfy memory bound along with training computation reduction, with notably higher accuracy compared with directly training a shrinked-size DNN model. In order to obtain such merits, a sparse mask (i.e., a set of binary values that has the same shape as weights and contains only 0 or 1) needs to be developed, in which the 1s activate a portion of the original dense network by selecting weights (i.e., usually less than 30% of the total weights) for training and inference. Under such circumstances, finding the binary mask before or during training surges as the key enabler to provide accurate and efficient learning dynamics (Lee et al., 2019; Wang et al., 2020b; Mocanu et al., 2018). Among a variety of methods, the Dynamic Sparse Training (DST) (Mocanu et al., 2018; Yin et al., 2024; Evci et al., 2020; Yuan et al., 2021; Li et al., 2023b; Dai et al., 2023) has emerged with the ability to optimize weights and find the mask concurrently, and rapidly becomes the prevailing approach owing to its high accuracy, efficiency, and ease of implementation.

In prior DST algorithms, a sparse mask keeps the entire training phase following an "always sparse" regime, and is

[*]Equal contribution [1]Clemson University, USA [2]University of Aberdeen, Scotland [3]University of Georgia, USA [4]University of Oxford, England. Correspondence to: Jie Ji <jji@g.clemson.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

evolving its topology by iteratively pruning weights with smallest magnitude and then growing other weights back. As shown in Figure 1 (a), during training, the updates on model weights and masks form a bi-level structure: ① depending on the magnitude of the updated weights, the topology of the mask is then updated, which forms the upper-level problem, ② depending on the newly obtained masks, the weights of the subnetwork will be updated through the standard SGD process (Amari, 1993), which forms the lower-level problem. Both problems ① and ② are mutually dependent on the outcome of one another, which indicates that the formation of the DST essentially falls within the bi-level optimization (Colson et al., 2005; Zhang et al., 2023) realm, and both should be solved analytically. However, current DST optimizes the weight objective with conventional SGD optimization, while leaving the mask objective updated using non-comprehensive heuristics (e.g., weight magnitude (Mocanu et al., 2018; Mostafa & Wang, 2019; Yuan et al., 2021; Liu et al., 2021; Yuan et al., 2022), gradient (Evci et al., 2020; Jayakumar et al., 2020; Schwarz et al., 2021), momentum (Dettmers & Zettlemoyer, 2019), etc.). Consequently, the search efficiency for the mask is relatively low, making it prone to sub-optimal training accuracy. For instance, the mask tends to *stop evolving early* due to the convergence of weights (Yuan et al., 2021). Increasing mask update frequency along with using large learning rate and batch size (Evci et al., 2020) may partially alleviate the issue but incurs high system overhead (i.e., the amount of time or resources required in addition to those for computation in an end-to-end training process) (Chen et al., 2023a). Additionally, heuristic mask updating loses the opportunity for exploring the topological space of a neural network where the *learned structure* contains latent expressiveness power (Ramanujan et al., 2020). Although recent works have observed the performance gain of subnetworks through optimization (Tai et al., 2022), studies of their adoption in the DST methods are still lacking.

In this paper, we scrutinize the optimization opportunities in every phase of the DST, and we argue that the training scheme of DST essentially forms a bi-level structure, in which both levels should be solved simultaneously through optimization techniques. As such, we present a novel framework **Bi**-level **D**ynamic **S**parse **T**raining (**BiDST**), which for the first time, matures the partial-heuristic DST scheme into a fully optimization-based methodology. In BiDST, as shown in Figure 1 (b), the mask is optimized based on both weight and its topology, which are intercorrelated via implicit gradient (Krantz & Parks, 2002). We show that our approach not only provides a theoretically-grounded solution towards sparse training, but also incurs only negligible extra computations that can be easily compensated through adjusting the training schedule without degrading accuracy. We demonstrate that BiDST shows superior mask searching quality and efficiency. Compared to prior DST methods, BiDST achieves notably higher accuracy while obtaining a more meaningful mask that can be used to identify a subnetwork with better training dynamics. More importantly, the optimization-based methodology serves as the driving force behind a more efficient mask searching procedure, which alleviates the system overhead problem that is primarily caused by frequently changing mask topology in DST, thus providing a more practical way to achieve efficient training. To summarize our contributions:

- We scrutinize every phase in the DST methods and find the lost treasure – an optimization opportunity on the mask update part that is long-neglected. This opportunity should be incorporated to reform DST into a bi-level structure, enabling comprehensive optimization.
- We design a novel sparse training framework – BiDST, which adopts bi-level optimization techniques to solve the weight and mask objective functions simultaneously while only incurring negligible computation costs.
- We demonstrate that BiDST excels in efficiently exploring the mask searching space, thus discovering superior sparse patterns that result in higher accuracy compared to other dynamic sparse training methods.
- We analyze that the frequent mask update is the source of the primary system overhead in the DST family, and demonstrate that the proposed BiDST framework is highly practical for the end-to-end implementation thanks to its optimization nature which reduces update frequency.

2. Proposed Method

2.1. Notations and Preliminaries

Consider a network function $f(\cdot)$ that is initialized as $f(x; \boldsymbol{\theta}_0)$ where x denotes input training samples. We use $\boldsymbol{\psi}$ to define a sparse mask $\boldsymbol{\psi} \in \{0,1\}^{|\boldsymbol{\theta}|}$ that is obtained from certain pruning algorithm, and a sparse subnetwork is defined as $(\boldsymbol{\psi} \odot \boldsymbol{\theta})$ where $\odot$ is the element-wise multiplication. We use s to denote the sparsity ratio, which is defined as the percentage of the *pruned* weights in the DNN model. For an entire training process, $t \in (1, T)$ denotes a certain iteration within the total T iterations. We use ℓ to characterize the loss of the function $f(\cdot)$. By minimizing $\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}))$, a learned sparse subnetwork can be written as $f(\boldsymbol{\psi}_T \odot \boldsymbol{\theta}_T)$, where x is omitted for the ease of notation purpose.

Dynamic sparse training (DST). To reduce the training computation cost and save hardware resources, DST trains a randomly initialized sparse neural network $f(\boldsymbol{\psi}_0 \odot \boldsymbol{\theta}_0)$ from scratch, and adjusts the sparsity topology $\boldsymbol{\psi}$ during training.

The training scheme of a classic DST can be summarized as a process of alternating the following two phases:

- *Weight training:* based on the developed mask $\boldsymbol{\psi}_{t-1}$, we train the network for t iterations, arriving at weights θ_t and network function $f(\boldsymbol{\psi}_{t-1} \odot \boldsymbol{\theta}_t)$.
- *Mask updating*: based on the trained weights $\boldsymbol{\theta}_t$, we use a mask updating function `MaskUpdate` to change the network topology to a better structure, i.e., $\boldsymbol{\psi}_t \leftarrow \texttt{MaskUpdate}(\boldsymbol{\psi}_{t-1}, \boldsymbol{\theta}_t)$.

In recent literature (Mostafa & Wang, 2019; Evci et al., 2020; Yuan et al., 2021), `MaskUpdate` is usually achieved by a two-step prune-grow process. First, the heuristic magnitude-based pruning prunes the weights to $(s+p)$, where p is a hyper-parameter that increases the pruning ratio to a higher level. Then, a new mask $\boldsymbol{\psi}$ is obtained by growing a number of $p \times |\boldsymbol{\theta}|$ zero weights back, resetting the sparsity level back to s. Formally, the `MaskUpdate` can be defined as

$$\begin{aligned} &f(\boldsymbol{\psi}_t \odot \boldsymbol{\theta}_t) | \boldsymbol{\psi}_t \leftarrow \texttt{ArgGrowTo}(Pt, s) \\ &\text{where } \; Pt = \texttt{ArgPruneTo}(f(\boldsymbol{\psi}_{t-1} \odot \boldsymbol{\theta}_t), s+p) \end{aligned} \tag{1}$$

Bi-level optimization (BO). Bi-level optimization (BO) serves as a valuable approach for modeling scenarios characterized by a two-tier hierarchy, wherein the upper-level variables exhibit a dependency on the outcomes derived from specific lower-level optimization problems. In general, a standard representation of BO can be expressed as

$$\underset{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}}{\text{minimize}} \; U(\boldsymbol{x}, \boldsymbol{v}^*(\boldsymbol{x})); \qquad \text{s.t. } \; \boldsymbol{v}^*(\boldsymbol{x}) \in \underset{\boldsymbol{v} \in \boldsymbol{\mathcal{V}}}{\arg\min} \; L(\boldsymbol{x}, \boldsymbol{v}), \tag{2}$$

where $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{V}}$ are the feasible sets for the upper-level variable $\boldsymbol{x}$ and the lower-level variable $\boldsymbol{v}$, respectively. Consequently, this two-tier hierarchy can be characterized by two objectives as

- *Upper-level objective:* $U(\cdot)$ represents the upper-level objective, which aims to find an optimal solution $\boldsymbol{x}$ by considering the impact of the lower-level problem.
- *Lower-level objective:* $L(\cdot)$ represents the lower-level objective, which involves finding an optimal $\boldsymbol{v}$ that minimizes $L(\boldsymbol{x}, \boldsymbol{v})$ given $\boldsymbol{x}$.

The two levels of optimization are interdependent, and the solution to the upper-level problem depends on the outcome of the lower-level problem.

Why dynamic sparse training is a bi-level optimization problem? From the above discussion, it is clear that the two phrases in the DST training scheme are interdependent, that the development of mask is based on the trained weights while the topology of the weights is characterized by the developed mask. This process is very much similar to the formulation of a BO problem in equation 2. Considering the goal of DST is to train the weights while obtaining a good sparse topology (i.e., mask), the previous DST scheme can be informally written as

$$\begin{aligned} &\texttt{MaskUpdate}(f(\boldsymbol{\psi}_{t-1} \odot \boldsymbol{\theta}_t)); \\ &\text{s.t. } \; \boldsymbol{\theta}_t \in \arg\min \; \ell(f(\boldsymbol{\psi}_{t-1} \odot \boldsymbol{\theta}_{t-1})), \forall \, t \in (0, T), \end{aligned} \tag{3}$$

where the dependency between $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$ is characterized by how the solutions are obtained given certain conditions, i.e., given $\boldsymbol{\theta}_t$, $\texttt{MaskUpdate}(\cdot)$ finds $\boldsymbol{\psi}_t$, while $\boldsymbol{\theta}_t$ is trained from $\boldsymbol{\theta}_{t-1}$ that is structured by $\boldsymbol{\psi}_{t-1}$. It can be inferred that upper-level objective in equation 2 can be extended to $U(\boldsymbol{x}, \boldsymbol{v}^*(\boldsymbol{x})) \leftarrow \texttt{MaskUpdate}(f(\boldsymbol{\psi}_{t-1} \odot \boldsymbol{\theta}_t))$, and the lower-level objective in equation 2 can be extended to $L(\boldsymbol{x}, \boldsymbol{v}) \leftarrow \ell(f(\boldsymbol{\psi}_{t-1} \odot \boldsymbol{\theta}_{t-1}))$. Therefore, equation 2 is symbolically equivalent to equation 3, thus DST is essentially a bi-level optimization problem.

2.2. BiDST - Bi-Level Dynamic Sparse Training Framework

The intuition in Section 2.1 asserts that the DST is a BO problem, particularly because of the strong resemblance to the structure and characteristics of the latter one. In equation 3, we can see that the current mask $\boldsymbol{\psi}_t$ is developed from $\boldsymbol{\psi}_{t-1}$ using current weights $\boldsymbol{\theta}_t$, and in the meantime, $\boldsymbol{\theta}_t$ is trained from $\boldsymbol{\theta}_{t-1}$ with previous topology guided by $\boldsymbol{\psi}_{t-1}$. The two phrases are interdependent on each other, where the mask determines which portions of the weights are subjected to training, and the weights, in turn, influence the subsequent sparse topology. Concretely, the solution to the upper-level problem in equation 3 is given by equation 1, which usually relies on rule-based heuristics that lead to sub-optimal solutions. Inspired by the fact that mask is embedded in the network computation graph due to its association with weights, we find that mask is implicitly characterized by a single unified loss function, and can be optimized through gradient-based techniques. Therefore, we rewrite equation 3 into the following bi-level dynamic sparse training (BiDST) formulation:

$$\begin{aligned} \underset{\boldsymbol{\psi} \in \boldsymbol{\Psi}}{\text{minimize}} \quad & \underbrace{\ell(f(\boldsymbol{\psi} \odot \overbrace{\boldsymbol{\theta}^*(\boldsymbol{\psi})}^{\text{Trained } \boldsymbol{\theta}}))}_{\text{Loss of the mask}}; \\ \text{s.t.} \quad & \underbrace{\boldsymbol{\theta}^*(\boldsymbol{\psi}) \in \underset{\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}}{\arg\min} \; \ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta})) + \frac{\lambda}{2} \|\boldsymbol{\psi} \odot \boldsymbol{\theta}\|^2}_{\text{Optimize weight } \boldsymbol{\theta} \text{ given fixed mask } \boldsymbol{\psi}}, \end{aligned} \tag{4}$$

where ℓ is the cross-entropy loss, and the regularization term in the lower-level problem with hyper-parameter λ stabilizes the gradient flow (Shaham et al., 2015; Hong et al.,

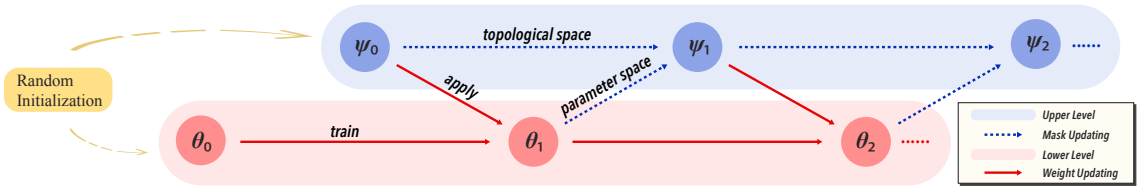


Figure 2: Overview of the two-level dynamic mask updating scheme of BiDST. The upper level learns the mask using topological space information from the previous mask and parameter space information from weights, and the lower level trains the sparse weights guided by mask.

2020). Depending on different objective variables, equation 4 decomposes a DST problem into two sub-problems:

- *Upper-level problem:* given trained weights $\boldsymbol{\theta}$, the upper-level problem aims to find a better mask $\boldsymbol{\psi}$ by minimizing the loss in terms of the mask.
- *Lower-level problem:* given a fixed mask $\boldsymbol{\psi}$, the lower-level problem updates the subnetwork weights by minimizing the loss in terms of weights, which is usually achieved by SGD.

By changing the upper-level objective from `MaskUpdate`$(\cdot)$ to the loss function of the network, BiDST exhibits better mask search ability by systematically exploring the solution space. Comparing to the trial-and-error strategies used in prior DST methods, BiDST can find globally near-optimal solutions more efficiently. In BiDST, a single loss function couples the topological space (i.e., mask) and the parameter space (i.e., weights) for joint optimization (see Figure 1(b) and Figure 2), which leverages the latent expressiveness power of the *learned structure* of the neural network. Additionally, the proposed optimization approach only incurs negligible computations by reusing gradients in the computation graph, and can be practically implemented by using Gumbel-Softmax (Jang et al., 2017) (please see Section 2.3).

To solve the BiDST optimization problem, we apply chain rule on the mask loss function $\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi})))$, we have:

$$\begin{aligned}\frac{d\ell(f(\boldsymbol{\psi}\odot\boldsymbol{\theta}^*(\boldsymbol{\psi})))}{d\boldsymbol{\psi}} &= \nabla_{\boldsymbol{\psi}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi}))) \\ &+ \frac{d\boldsymbol{\theta}^*(\boldsymbol{\psi})}{d\boldsymbol{\psi}}^\top \nabla_{\boldsymbol{\theta}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi}))),\end{aligned} \tag{5}$$

where $\frac{d\ell(f(\boldsymbol{\psi}\odot\boldsymbol{\theta}^*(\boldsymbol{\psi})))}{d\boldsymbol{\psi}}$ denotes the full derivative of ℓ, $\nabla_{\boldsymbol{\psi}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi})))$ and $\nabla_{\boldsymbol{\theta}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi})))$ denote the partial derivative of the loss function with respect to mask and weights, respectively. Therefore, the optimization of the upper-level problem jointly considers both the topological space ($\nabla_{\boldsymbol{\psi}}$) and the parameter space ($\nabla_{\boldsymbol{\theta}}$).

According to the implicit function theory (Gould et al., 2016), we refer to $\frac{d\boldsymbol{\theta}^*(\boldsymbol{\psi})}{d\boldsymbol{\psi}}^\top$ as implicit gradient (IG). For ease of notations, the transpose operation $\top$ is omitted in the rest of the paper. In equation 5, with the first-order stationary condition of BO (Ghadimi & Wang, 2018), we have

$$\nabla_{\boldsymbol{\psi}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi}))) + \lambda\boldsymbol{\theta}^* = 0. \tag{6}$$

By taking derivative with respect to $\boldsymbol{\psi}$ on both sides, we have:

$$\begin{aligned}\nabla^2_{\boldsymbol{\psi}\boldsymbol{\theta}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi}))) &+ \frac{d\boldsymbol{\theta}^*(\boldsymbol{\psi})}{d\boldsymbol{\psi}}\nabla^2_{\boldsymbol{\theta}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi}))) \\ &+ \lambda\frac{d\boldsymbol{\theta}^*(\boldsymbol{\psi})}{d\boldsymbol{\psi}} = 0.\end{aligned} \tag{7}$$

So that we can get

$$\frac{d\boldsymbol{\theta}^*(\boldsymbol{\psi})}{d\boldsymbol{\psi}} = -\nabla^2_{\boldsymbol{\psi}\boldsymbol{\theta}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi})))(\nabla^2_{\boldsymbol{\theta}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi}))) + \lambda\mathbf{I})^{-1}. \tag{8}$$

Since directly computing the second-order partial derivatives is theoretically challenging, inspired by Finn et al. (2017) and Liu et al. (2022a), we replace $\nabla^2_{\boldsymbol{\theta}}$ with zero and therefore

$$\begin{aligned}\frac{d\ell(f(\boldsymbol{\psi}\odot\boldsymbol{\theta}^*(\boldsymbol{\psi})))}{d\boldsymbol{\psi}} &= \nabla_{\boldsymbol{\psi}}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi}))) \\ &- \frac{1}{\lambda}\nabla_{\boldsymbol{\psi}\odot\boldsymbol{\theta}^*(\boldsymbol{\psi})}\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}^*(\boldsymbol{\psi}))).\end{aligned} \tag{9}$$

At this point, solving the upper-level problem for mask update is to compute the first-order partial derivatives. Based on Hong et al. (2023), the analytical solution for the BiDST defined in equation 4 is derived following the classic DST regime in Section 2.1 as

- *Weight training:* at iteration t, we solve the *lower-level* problem using SGD, which is given by

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha\nabla_{\boldsymbol{\theta}}[\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta})) + \frac{\lambda}{2}\|\boldsymbol{\psi} \odot \boldsymbol{\theta}\|^2]_{t-1}. \tag{10}$$

- *Mask updating*: after obtaining $\boldsymbol{\theta}_t$, we solve the *upper-level* problem with

$$\boldsymbol{\psi}_t = \boldsymbol{\psi}_{t-1} - \alpha\frac{d\ell(f(\boldsymbol{\psi} \odot \boldsymbol{\theta}_t))}{d\boldsymbol{\psi}}\Big|_{\boldsymbol{\psi}=\boldsymbol{\psi}_{t-1}} \tag{11}$$

Table 1: Accuracy comparison using ResNet-32 on CIFAR-10/100.

Datasets	Sparsity Distribution	CIFAR-10 (*dense: 94.9*)			CIFAR-100 (*dense: 74.9*)		
Pruning ratio		90%	95%	98%	90%	95%	98%
LT [13]	non-uniform	92.31	91.06	88.78	68.99	65.02	57.37
SNIP [26]	non-uniform	92.59	91.01	87.51	68.89	65.02	57.37
GraSP [51]	non-uniform	92.38	91.39	88.81	69.24	66.50	58.43
Deep-R [2]	non-uniform	91.62	89.84	86.45	66.78	63.90	58.47
SET [35]	non-uniform	92.30	90.76	88.29	69.66	67.41	62.25
DSR [38]	non-uniform	92.97	91.61	88.46	69.63	68.20	61.24
RigL 11	ERK	93.55	92.39	90.22	70.62	68.47	64.14
RigL-ITOP 32	ERK	93.70	92.78	90.40	71.16	69.38	66.35
BiDST (ours)	ERK	**94.12±0.11**	**93.26±0.12**	**92.21±0.09**	**73.42±0.14**	**71.50±0.07**	**68.19 ±0.15**
RigL-ITOP [32]	uniform	93.19	92.08	89.36	70.46	68.39	64.16
RigL [11]	uniform	93.07	91.83	89.00	70.34	68.22	64.07
MEST [54]	uniform	92.56	91.15	89.22	70.44	68.43	64.59
BiDST (ours)	uniform	**93.68±0.11**	**92.93±0.14**	**91.99±0.12**	**72.97±0.08**	**70.03±0.12**	**67.21 ±0.06**

Algorithm 1 BiDST implementation details

Input: A DNN model with randomly initialized weight $\boldsymbol{\theta}_0$; a random mask $\boldsymbol{\psi}_0$ with sparsity s. A flag parameter $\mathcal{F}$ indicating when to change the subnetwork topology based on $\boldsymbol{\psi}$.

Output: A sparse model satisfying the target sparsity s.

Set $t = 0$. Set the number of non-zero weights to be $k = s \times |\boldsymbol{\theta}|$.

while $t < T$ **do**

  if $\mathcal{F}$ **then**

    $\boldsymbol{\psi}'_t \leftarrow \texttt{Binarize}(\boldsymbol{\psi}_t, \texttt{argmax}(\boldsymbol{\psi}_t, k))$

  Train the subnetwork $f(\boldsymbol{\psi}'_t \odot \boldsymbol{\theta}_t)$ by solving Eq. 10.

  Update mask $\boldsymbol{\psi}$ by solving Eq. 11.

  $t = t + 1$.

where α is the learning rate. We set α for upper- and lower-level solution to be the same for easy implementation. Algorithm 1 briefly shows the training process of the proposed BiDST.

2.3. Computation Analysis for BiDST

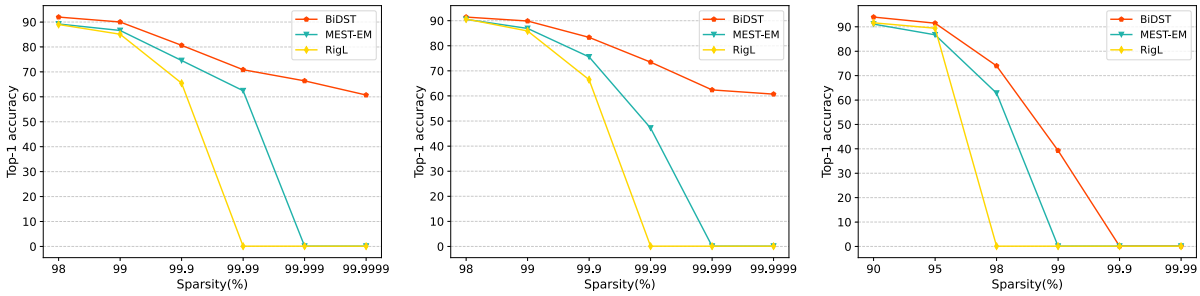
In BiDST, the key parameters within the computation graph (e.g., weight, mask) are performing sparse computation (i.e., in both forward and backward propagation) to reduce the training cost. From Algorithm 1, weight and mask are two components that majorly contribute to the computation budget. Compared to prior DST methods, BiDST relaxes the mask variable to a continuous variable and computes its gradients, thus involving extra computation that arises from optimizing the mask. To minimize such extra cost, we apply `argmax(·)` function on mask to index a subset of weights for forward propagation. During backward propagation, mask gradients are derived from the same copy of activation gradients that is used to compute weight gradients. Therefore, mask doesn't take any input for forward computation, and we certainly do not need to re-compute activation gradients when applying chain rule. Specifically, BiDST only learns a subset of mask that covers the current DNN topology and an extra space for mask updating (i.e., please see Appendix A for settings and discussion), which not only guarantees sparse computation on regular DNN computation, but also ensures sufficient search space (Lee et al., 2014) for mask updating.

2.4. Overhead Reduction for Practical Implementation

Unlike other approaches that frequently change mask topology in their DST schedule (Evci et al., 2020; Mocanu et al., 2018), BiDST learns the subset of relaxed mask continuously but only changes the network topology periodically (i.e., `Binarize` through `argmax` operation of mask, please refer to $\mathcal{F}$ in Algorithm 1 and its setting in Appendix A). In practice, frequently changing the network topology incurs huge system overhead, which means more time and hardware resources are required to perform end-to-end learning due to computation graph reconstruction and static machine code recompilation (Chen et al., 2018). We show that BiDST only requires limited topology updates thanks to more principled and data-driven solutions (please see results in Section 3.4), which significantly reduces overhead and promotes practical implementations.

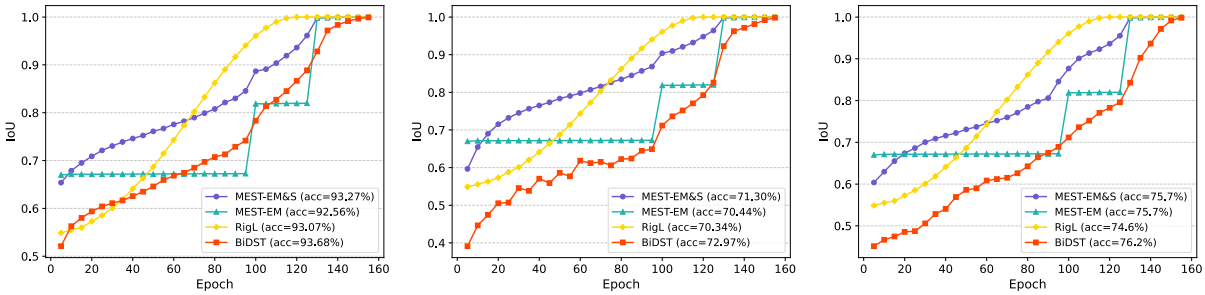
3. Experimental Results

In this section, we carry out experiments to comprehensively demonstrate the advantages of BiDST. We evaluate BiDST in comparison with the state-of-the-art (SOTA) DST methods, and show superior accuracy, effective mask searching ability, as well as great applicability for implementations. We follow the traditional network and dataset selection used in prior DST methods. We use ResNet-32 (Zagoruyko & Komodakis, 2016), VGG-19 (Simonyan & Zisserman, 2014) and MobileNet-v2 (Sandler et al., 2018) on CIFAR-



(a) ResNet-32 CIFAR-10 (b) VGG-19 CIFAR-10 (c) MobileNet-v2 CIFAR-10

Figure 3: We perform high sparsity experiments on ResNet-32, VGG-19 and MobileNet-v2 on CIFAR-10 dataset, and compare with different DST methods. Note that MobileNet-v2 is already a compact model, so the extreme sparsity range is different from the other two networks.



(a) ResNet-32 CIFAR-10 (b) ResNet-32 CIFAR-100 (c) ResNet-50 ImageNet-1K

Figure 4: Evaluation of BiDST mask development progress. We use ResNet-32 on CIFAR-10/100, and ResNet-50 on ImageNet-1K at a target sparsity of 90%.

10 and CIFAR-100 datasets (Krizhevsky, 2009), and we use ResNet-34 and ResNet-50 (He et al., 2016) on ImageNet-1K dataset (Deng et al., 2009). We test the on-device training performance using a Samsung Galaxy S21 with Snapdragon 888 chipset. We repeat training experiments for 3 times for all experiments.

3.1. Experimental Settings

We use standard training recipe following Yuan et al. (2021) and Wang et al. (2020b). To ensure fair comparison, all BiDST experiments have a slight scale down on the number of training epochs to compensate the mask learning computation cost. We use standard data augmentation, and cosine annealing learning rate schedule is used with SGD optimizer. For CIFAR-10/100, we use a batch size of 64 and set the initial learning rate to 0.1. For ImageNet-1K, we use a batch size of 1024 and learning rate of 1.024 with a linear warp-up for 5 epochs. Due to limited space, we put detailed settings in Appendix A.

3.2. BiDST Accuracy

CIFAR-10 and CIFAR-100. We test BiDST using uniform and ERK (Evci et al., 2020) sparsity with 90%, 95%, and 98% overall sparsity. We demonstrate the accuracy with standard deviation results of ResNet-32 in Table 1. Due to space limitation, the results of all ERK sparsity, VGG-19 and MobileNet-v2 are included in Appendix B. According to the results, BiDST establishes a new state-of-the-art accuracy bar for various sparse training methods. When compared to state-of-the-art DST methods such as RigL and MEST, BiDST consistently outperforms the prior approaches by a significant margin. At varying levels of sparsity, we can see that BiDST achieves higher accuracy compared to the best baseline accuracy on CIFAR-10 by 0.49%, 1.1% and 2.62%, respectively, and on CIFAR-100 by 2.52%, 2.12% and 2.62%, respectively. With VGG-19 and MobileNet-v2, similar outperforming results can also be observed. Note that MobileNet-v2 is commonly considered a compact network that is hard to incorporate sparsity, BiDST still achieves promising results, which indicates that a more meaningful mask, though hard to find, can still be discovered by our method.

ImageNet-1K. Table 2 shows ResNet-50 on ImageNet-1K accuracy with uniform sparsity (ResNet-34 results in Appendix B). We stress all experiments are performed *three times*. We report the average accuracy and omit the standard deviation since the training is relatively sta-

Table 2: Accuracy comparison using ResNet-50 on ImageNet-1K with uniform sparsity.

Method	Top-1 Accuracy (%)	Training FLOPS (×e18)	Inference FLOPS (×e9)	# of Mask Updates	Top-1 Accuracy (%)	Training FLOPS (×e18)	Inference FLOPS (×e9)	# of Mask Updates
Dense	77.1	4.8	8.2	n/a	-	-	-	-
Sparsity ratio	80%				90%			
SNIP [26]	69.7	1.67	2.8	Static	62.0	0.91	1.9	Static
GraSP [50]	72.1	1.67	2.8	Static	68.1	0.91	1.9	Static
DeepR [2]	71.7	n/a	n/a	n/a	70.2	n/a	n/a	n/a
SNFS [10]	73.8	n/a	n/a	n/a	72.3	n/a	n/a	n/a
DSR [38]	73.3	1.28	3.3	n/a	71.6	0.96	2.5	n/a
SET [35]	72.6	0.74	1.7	n/a	70.4	0.32	0.9	n/a
RigL [11]	74.6	0.74	1.7	320	72.0	0.39	0.9	320
$\text{MEST}_{0.67\times}$ [54]	75.4	0.74	1.7	45	72.6	0.39	0.9	45
SpFDE [55]	75.4	0.74	1.7	45	-	-	-	-
$\text{BiDST}_{0.67\times}$ (ours)	**75.8**	0.74	1.7	12	**72.9**	0.39	0.9	12
MEST [54]	75.7	1.27	1.7	60	75.0	0.65	0.9	60
$\text{RigL}_{2\times}$*	75.5	1.49	1.7	640	74.6	0.78	0.9	640
Top-KAST [22]	-	-	-	-	73.0	0.63	0.9	320
BiDST (ours)	**76.1**	1.26	1.7	18	**75.3**	0.65	0.9	18
$\text{MEST}_{1.7\times}$	76.7	1.84	1.7	110	75.9	0.80	0.9	110
$\text{RigL}_{5\times}$ [11]	76.6	3.65	1.7	1,600	75.7	1.95	0.9	1,600
$\text{BiDST}_{1.7\times}$ (ours)	**76.9**	1.84	1.7	31	**76.2**	0.88	0.9	31
$\text{MEST}_{8\times}$*	77.6	8.80	1.7	550	77.0	3.84	0.9	550
$\text{RigL}_{12\times}$*	77.4	8.90	1.7	3,840	76.8	4.69	0.9	3,840
C-GaP [34]	77.9	12.07	1.7	28	76.3	9.69	0.9	28
$\text{BiDST}_{8\times}$ (ours)	**77.9**	8.80	1.7	146	**77.3**	3.85	0.9	146

* Our implementation with original source code using long training time.

ble (i.e., usually less than 0.1% standard deviation) on large-scale datasets. Once again, BiDST achieves dominating performance against all baseline methods with regular training recipe. At similar training FLOPs, BiDST achieves significantly higher accuracy and requires fewer mask updates. When increasing the training time, BiDST exhibits promising accuracy gain. We extend training epochs to match $\text{RigL}_{2\times}$ (200 epochs), $\text{MEST}_{1.7\times}$ (250 epochs), $\text{RigL}_{5\times}$ (500 epochs), C-GaP (Ma et al., 2022) (990 epochs), $\text{MEST}_{8\times}$ and $\text{RigL}_{12\times}$ (1200 epochs). Note that C-GaP has partially-dense model during training, which has notably higher training FLOPs.

Extreme Sparsity. In dynamic sparse training, layer-wise sparsity ratios are pre-defined (e.g., uniform or ERK) to avoid layer collapse (Tanaka et al., 2020) in high sparsity (i.e., an entire layer is pruned). Nonetheless, different layers still experience different learning dynamics (Chen et al., 2023b), and high sparsity may prevent gradient from flowing smoothly. We conduct extreme sparsity ratios (e.g., up to $s = 99.9999\%$) on BiDST and compare the results to different DST methods. As shown in Figure 3, BiDST demonstrates better stability in extreme sparsity, indicating that a more accurate mask can be identified from the network topology in any condition, such as large sparsity settings in training or when sparsifying a compact network structure (e.g., MobileNet family).

3.3. Sparse Mask Evaluation and Analysis

The optimization-based BiDST achieves higher mask searching efficiency compared to the prior heuristic-based approaches. For instance, the magnitude-based pruning and gradient (momentum) -based growing methods for mask updating may be significantly affected by the convergence of weights with the magnitude of the parameters oscillating within a very small range after certain times of training. Therefore, the commonly-used hard thresholding for such methods may not work effectively since the fluctuation is not sufficient enough to exceed certain thresholds. In BiDST, the mask updating is determined by jointly considering the importance of *both* network topology *and* weight value (i.e., please refer to Figure 2). We use Intersection over Union (IoU) to describe the development of mask during training process. As shown in Figure 4, at the same sparsity and within the same searching space, we compute the mask IoU of fixed intervals between two mask updates, and the results show that BiDST achieves better mask development (i.e., lower IoU means a higher portion of mask is updated). Compared to baseline methods that either suffer from high IoU or manually enforce the values within a certain range (e.g., MEST-EM&S), BiDST demonstrates a sufficient searching space and natural mask learning curves.

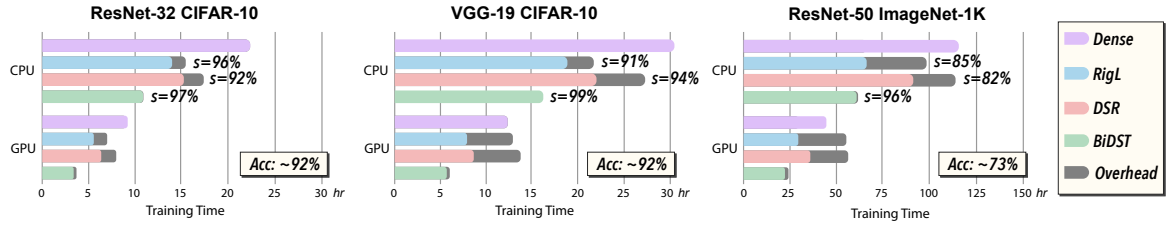


Figure 5: System overhead in DST when implemented on Snapdragon 888. We compare representative DST methods (e.g., uniform sparsity RigL (Evci et al., 2020), global reparameterization DSR (Mostafa & Wang, 2019)) with BiDST in uniform sparsity at similar accuracy.

3.4. Training Speedup and System Overhead of Dynamic Sparse Training

To demonstrate practical implementation of BiDST, we extend the code generation of TVM (Chen et al., 2018) and design a training engine on the Snapdragon 888. For DST, the on-device computation is done by static code (e.g., OpenCL for GPU and C++ for CPU), and the training acceleration is obtained from compiler optimization that skips the zeros in weights. We set a target accuracy and perform experiments using representative DST methods. According to Figure 5, BiDST achieves the best training acceleration performance (i.e., highest sparsity) among all other methods. Compared to dense training, BiDST achieves training speedup with ResNet-32-CIFAR10 (CPU: 2.01×, GPU: 2.14×), VGG19-CIFAR10 (CPU: 1.88×, GPU: 2.04×), and ResNet-50-ImageNet (CPU: 1.88×, GPU: 1.83×), respectively. Meanwhile, dynamic topology switching on the hardware requires code re-compilation, which is the source of the major system overhead in DST. The mask update frequency of BiDST is the lowest among other methods, resulting in 16-25× reduced overhead time. Additionally, the uniform sparsity of BiDST alleviates the resource allocation burden, which further contributes to better speedup compared to the global reparameterization method such as DSR (Mostafa & Wang, 2019).

4. Related Works

Dynamic sparse training. Parameter importance estimation plays a pivotal role in DST, aiding in the identification of parameters worthy of retention or pruning. Various heuristic-based methods are employed for this purpose. For example, SET (Mocanu et al., 2018) removes the least magnitude-valued weights during training and regrows an equivalent number of weights randomly at the end of each epoch. SNFS (Dettmers & Zettlemoyer, 2019) uses exponentially smoothed momentum to identify important weights and layers, redistributing pruned weights based on mean momentum magnitude. RigL (Evci et al., 2020) updates the sparsity topology of the network during training using magnitude-based weight dropping method and regrows weights using the top-k absolute largest gradients. ITOP (Liu et al., 2021) highlights the benefits of dynamic mask training, considering all possible parameters across time. Top-KAST (Jayakumar et al., 2020) offers a scalable and consistently sparse DST framework for improved effectiveness and efficiency. MEST (Yuan et al., 2021) employs a gradually decreasing drop and grow rate with a more relaxed range of parameters for growing. AD/AC (Peste et al., 2021) proposes a co-training of dense and sparse models method, which generates accurate sparse–dense model pairs at the end of the training process. ToST(Jaiswal et al., 2022) provide a plug and play toolkit to find a better sparse mask for current pruning method. (Nowak et al., 2023) proposes that pruning criterion affects the update of network typology, thus impacting the final performance.

Heuristic- and Optimization-based Sparsity Learning. There are various heuristic-based methods for learning a sparse network, such as (Han et al., 2015; Molchanov et al., 2017; 2019; Frankle & Carbin, 2018). To improve the performance, data-driven approaches emerged to better optimize the mask finding process. SNIP (Lee et al., 2019) sparsifies the network at early stage of training by studying gradients of the training loss at initialization. GraSP (Wang et al., 2020b) preserves the gradient flow and prunes the network at initialization. SynFlow (Tanaka et al., 2020) studies the layer-collapse issue in sparse training and propose to use network synaptic flow to guide early pruning of the network. Additionally, numerous studies incorporate optimization techniques. ADMM (Zhang et al., 2018; Ren et al., 2019) transfers the non-convex optimization problem into two subproblems that are solved iteratively. OLMP (Li et al., 2018) transforms the threshold tuning problem into a constrained optimization problem and employs powerful derivative-free optimization algorithms to solve it. Spartan (Tai et al., 2022) combines soft masking and dual averaging-based updates, enabling flexible sparsity allocation policies. Liu et al. (2022b) uses the energy estimate of each layer as the Frobenius norm optimization constraint for the convolution kernel weights.

5. Conclusion

In this paper, we establish a full optimization-based dynamic sparse training framework BiDST, which reformulates the prior partial-heuristic-based dynamic sparse training into

a bi-level optimization problem, and solves the problem of each level (e.g., mask finding and weight training) simultaneously and analytically. BiDST achieves prominent accuracy in the parameter-sparse training regime due to a more principled and data-driven approach, while promoting great applicability for on-device implementation by reducing system overhead brought by dynamism in DST methods.

Acknowledgment

This work is partly supported by the National Science Foundation CCF-2312616, CCF-2008049, NASA 80NSSC23K1393, and the Army Research Office W911NF-24-1-0044. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the NSF, NASA, Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Impact Statement

The advocate of integrating bi-level optimization into dynamic sparse training domain will revolutionize most of the current efficient training frameworks deployed in edge computing systems. The most significant impact of this paper is to extend the knowledge of bi-level optimization theory into the realm of sparse network computing to proactively address the heuristic nature of the previous methods. The success of the proposed BiDST will inspire novel theoretical and systematical integration in emerging areas such as IoT in smart city, next-generation computing systems, and advanced edge computing paradigm, therefore driving both theoretical progress and tangible benefits. The research, characterized by its scientific nature, is not anticipated to engender any negative societal consequences.

References

Amari, S.-i. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

Bellec, G., Kappel, D., Maass, W., and Legenstein, R. Deep rewiring: Training very sparse deep net-works. In *International Conference on Learning Representations (ICLR)*, 2018.

Chen, S., Wei, S., Liu, C., and Yang, W. Dycl: Dynamic neural network compilation via program rewriting and graph optimization. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 614–626, 2023a.

Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., Guestrin, C., and Krishnamurthy, A. Tvm: An automated end-to-end optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 578–594, 2018.

Chen, Y., Yuille, A., and Zhou, Z. Which layer is learning faster? a systematic exploration of layer-wise convergence rate for deep neural networks. In *The Eleventh International Conference on Learning Representations*, 2023b.

Colson, B., Marcotte, P., and Savard, G. Bilevel programming: A survey. *4or*, 3:87–107, 2005.

Dai, Y., Li, G., Luo, F., Ma, X., and Wu, Y. Coupling fairness and pruning in a single run: a bi-level optimization perspective. *arXiv preprint arXiv:2312.10181*, 2023.

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

Deng, L., Li, G., Han, S., Shi, L., and Xie, Y. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108 (4):485–532, 2020.

Dettmers, T. and Zettlemoyer, L. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 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. PMLR, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR*, 2018.

Ghadimi, S. and Wang, M. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hong, M., Wai, H., Wang, Z., and Yang, Z. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. arxiv e-prints, art. *arXiv preprint arXiv:2007.05170*, 2020.

Hong, M., Wai, H.-T., Wang, Z., and Yang, Z. A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic. *SIAM Journal on Optimization*, 33(1):147–180, 2023.

Jaiswal, A. K., Ma, H., Chen, T., Ding, Y., and Wang, Z. Training your sparse neural network better with any mask. In *International Conference on Machine Learning*, pp. 9833–9844. PMLR, 2022.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Jayakumar, S., Pascanu, R., Rae, J., Osindero, S., and Elsen, E. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33:20744–20754, 2020.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Krantz, S. G. and Parks, H. R. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Lee, N., Ajanthan, T., and Torr, P. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*, 2019.

Lee, S., Kim, J. K., Zheng, X., Ho, Q., Gibson, G. A., and Xing, E. P. On model parallelization and scheduling strategies for distributed machine learning. *Advances in neural information processing systems*, 27, 2014.

Li, G., Qian, C., Jiang, C., Lu, X., and Tang, K. Optimization based layer-wise magnitude-based pruning for dnn compression. In *IJCAI*, pp. 2383–2389, 2018.

Li, G., Ji, J., Qin, M., Niu, W., Ren, B., Afghah, F., Guo, L., and Ma, X. Towards high-quality and efficient video super-resolution via spatial-temporal data overfitting. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10259–10269. IEEE, 2023a.

Li, G., Yin, L., Ji, J., Niu, W., Qin, M., Ren, B., Guo, L., Liu, S., and Ma, X. Neurrev: Train better sparse neural network practically via neuron revitalization. In *The Twelfth International Conference on Learning Representations*, 2023b.

Liu, B., Ye, M., Wright, S., Stone, P., and qiang liu. BOME! bilevel optimization made easy: A simple first-order approach. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022a.

Liu, S., Yin, L., Mocanu, D. C., and Pechenizkiy, M. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pp. 6989–7000. PMLR, 2021.

Liu, S., Li, Q., Peng, B., Qiao, Y., Lin, M., and Zeng, Q. An energy-constrained optimization-based structured pruning method for deep neural network compression. In *2022 IEEE 22nd International Conference on Communication Technology (ICCT)*, pp. 1903–1907. IEEE, 2022b.

Ma, X., Qin, M., Sun, F., Hou, Z., Yuan, K., Xu, Y., Wang, Y., Chen, Y.-K., Jin, R., and Xie, Y. Effective model sparsification by scheduled grow-and-prune methods. In *International Conference on Learning Representations (ICLR)*, 2022.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9 (1):1–12, 2018.

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

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., and Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11264–11272, 2019.

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. PMLR, 2019.

Nowak, A. I., Grooten, B., Mocanu, D. C., and Tabor, J. Fantastic weights and how to find them: Where to prune in dynamic sparse training. *arXiv preprint arXiv:2306.12230*, 2023.

Peste, A., Iofinova, E., Vladu, A., and Alistarh, D. Ac/dc: Alternating compressed/decompressed training of deep neural networks. *Advances in neural information processing systems*, 34:8557–8570, 2021.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11893–11902, 2020.

Ren, A., Zhang, T., Ye, S., Li, J., Xu, W., Qian, X., Lin, X., and Wang, Y. Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 925–938, 2019.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Schwarz, J., Jayakumar, S., Pascanu, R., Latham, P. E., and Teh, Y. Powerpropagation: A sparsity inducing weight reparameterisation. *Advances in neural information processing systems*, 34:28889–28903, 2021.

Shaham, U., Yamada, Y., and Negahban, S. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Tai, K. S., Tian, T., and Lim, S. N. Spartan: Differentiable sparsity via regularized transportation. *Advances in Neural Information Processing Systems*, 35:4189–4202, 2022.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations (ICLR)*, 2020a.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020b.

Yin, L., Wu, Y., Zhang, Z., Hsieh, C.-Y., Wang, Y., Jia, Y., Pechenizkiy, M., Liang, Y., Wang, Z., and Liu, S. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*, 2023.

Yin, L., Li, G., Fang, M., Shen, L., Huang, T., Wang, Z., Menkovski, V., Ma, X., Pechenizkiy, M., Liu, S., et al. Dynamic sparsity is channel-level sparsity learner. *Advances in Neural Information Processing Systems*, 36, 2024.

Yuan, G., Ma, X., Niu, W., Li, Z., Kong, Z., Liu, N., Gong, Y., Zhan, Z., He, C., Jin, Q., Wang, S., Qin, M., Ren, B., Wang, Y., Liu, S., and Lin, X. Mest: Accurate and fast memory-economic sparse training framework on the edge. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Yuan, G., Li, Y., Li, S., Kong, Z., Tulyakov, S., Tang, X., Wang, Y., and Ren, J. Layer freezing & data sieving: Missing pieces of a generic framework for sparse training. *Advances in Neural Information Processing Systems*, 35: 19061–19074, 2022.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, T., Ye, S., Zhang, K., Tang, J., Wen, W., Fardad, M., and Wang, Y. A systematic DNN weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 184–199, 2018.

Zhang, Y., Khanduri, P., Tsaknakis, I., Yao, Y., Hong, M., and Liu, S. An introduction to bi-level optimization: Foundations and applications in signal processing and machine learning. *arXiv preprint arXiv:2308.00788*, 2023.

Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., and Zhu, J. Discrimination-aware channel pruning for deep neural networks. *Advances in neural information processing systems*, 31, 2018.

Appendix

A. Experiment Settings

Table A.1 lists training hyerparatemer settings for BiDST. For on-device speed evaluation, we scale down batch size to 64 for all network and dataset. Since we can use gradient accumulation to obtain larger batch size training, the accuracy for on-device experiments is not affected by different batch size. To make BiDST search efficiently, we use an extra space for mask to learn, which constitutes a very small portion of the mask beyond the space occupied by existing non-zero weights. We define this small portion of search space as $\mathcal{S}$, which is the ratio of the number of extra mask values to the total number of mask elements. During training, this small portion of mask is updating normally, but their corresponding weights and weight gradients are zeros. Therefore, this part of the mask is learning the importance of structure. Please note that mask learning in BiDST *does not* need any input, and reuses the activation gradient with Gumbel-Softmax for backward propagation, thus incurring very limited computation compared to traditional convolution operation. The epoch settings in Table A.1 can reflect how efficient our mask learning scheme is: we scale down the training time to match the training computation with other baselines, and the degree of epoch reduction is small compared to the total training epochs. For instance, our training epoch setting inherits MEST (Yuan et al., 2021) and GraSP (Wang et al., 2020a) as 160 epochs to train CIFAR-10/100, but we only train BiDST 158 epochs to compensate those extra computation of mask learning. For ImageNet-1K, we scale down training time from 150 epochs to 148 epochs to match computation with our comparison baselines.

B. Experimental Results Continue

B.1. CIFAR-10 and CIFAR-100 on VGG-19 and MobileNet-v2

We inlude the extra experimental results in this section. We demonstrate the results of VGG-19 and MobileNet-v2 on CIFAR-10/100, and both with uniform and ERK sparsity scheme. Table B.1 shows the VGG-19 results and Table B.2 shows the MobileNet-v2 results. As MobileNet-v2 is a compact network structure, we follow the classic settings (Zhuang et al., 2018) of training MobileNet-v2 and set the training epoch to 350 for better loss convergence.

B.2. ImageNet-1K on ResNet-34

Table B.3 shows the top-1 accuracy results of ResNet-34 on ImageNet-1K. For ImageNet-1K results, we use the uniform sparsity distribution. Similar to the settings in our main paper, we also extend the training time to $5\times$ training epochs. We can see from the results that the accuracy increase is consistently observed for $5\times$ setting, but not similarly significant compared to Resnet-50 network. We believe that the reason is that ResNet-34 is smaller than ResNet-50, thus the capacity of the network is not sufficient enough for fully capturing the complex patterns and features in the data.

C. Ablation Study

❖ **Mask search space $\mathcal{S}$ variation.** In BiDST, the mask search space is an extra space for mask to learn. The size of this space is related to the efficiency of the mask learning procedure. We manually set $\mathcal{S}$ with different value, and test the accuracy of ResNet-32 on CIFAR-10. We set the searching space ranging from 0.01 to 0.1, and show the results with different sparsity in Figure C.1 (a). As we can notice, the search space is not the larger the better. We choose to use the current setting because smaller space incurs less extra computation. We leave the research of BiDST search space optimization for future study.

❖ **Regularization coefficient λ.** We test BiDST at different regularization coefficient λ in Figure C.1 (b). λ controls the strength of a regularization term that discourages drastic changes in the mask by penalizing large gradients. A smaller λ allows for larger changes in the mask, while a larger λ enforces a stronger constraint on the changes to the mask. We set λ from 1e-7 to 1e-1 and perform BiDST with ResNet-32 on CIFAR-10. Please note that a larger change in mask update is not necessarily beneficial for accuracy (e.g., MEST-EM enforce strict search space). The high accuracy of BiDST is due to the optimization that systematically connects parameter space and topological space.

Table A.1: Hyperparameter settings.

Experiments	VGG-19	ResNet-32	MobileNet-v2	ResNet-50/34
Dataset	CIFAR	CIFAR	CIFAR	ImageNet-1K
Training hyperparameter settings				
Training epochs	158	158	347	148
Batch size	64	64	64	1024
Learning rate scheduler	cosine	cosine	cosine	cosine
Initial learning rate	0.1	0.1	0.1	1.024
Ending learning rate	4e-8	4e-8	4e-8	0
Momentum	0.9	0.9	0.9	0.875
ℓ_2 regularization	5e-4	1e-4	5e-5	3.05e-5
Warmup epochs	5	0	5	8
BiDST hyperparameter settings				
Mask update frequency	8	8	8	5
Mask search space ($\mathcal{S}$)	$s = 0.90, \mathcal{S} = 0.05$ $s = 0.95, \mathcal{S} = 0.03$ $s = 0.98, \mathcal{S} = 0.02$	$s = 0.90, \mathcal{S} = 0.05$ $s = 0.95, \mathcal{S} = 0.03$ $s = 0.98, \mathcal{S} = 0.02$	$s = 0.90, \mathcal{S} = 0.08$ $s = 0.95, \mathcal{S} = 0.05$ $s = 0.98, \mathcal{S} = 0.04$	$s = 0.90, \mathcal{S} = 0.05$ $s = 0.95, \mathcal{S} = 0.03$ $s = 0.98, \mathcal{S} = 0.02$
Regularization coefficient λ	1e-4	1e-4	1e-4	5e-5

Table B.1: Test accuracy of VGG-19 on CIFAR-10/100.

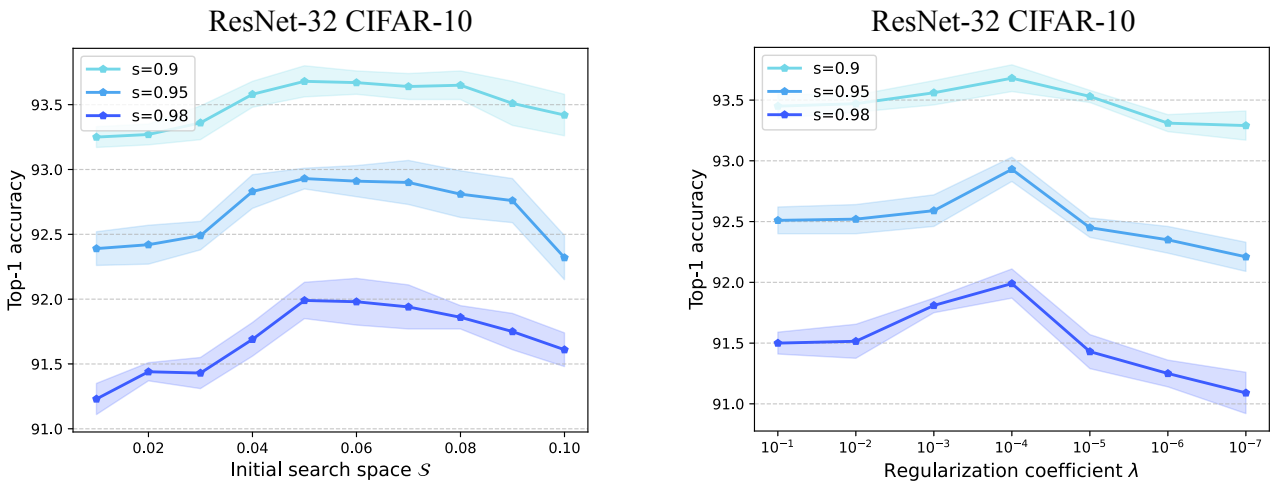
Methods	Sparsity Distribution	CIFAR-10 (*dense: 94.2*)			CIFAR-100 (*dense: 74.2*)		
Pruning ratio		90%	95%	98%	90%	95%	98%
LT [13]	non-uniform	93.51	92.92	92.34	72.78	71.44	68.95
SNIP [26]	non-uniform	93.63	93.43	92.05	72.84	71.83	58.46
GraSP [51]	non-uniform	93.30	93.43	92.19	71.95	71.23	68.90
Deep-R [2]	non-uniform	90.81	89.59	86.77	66.83	63.46	59.58
SET [35]	non-uniform	92.46	91.73	89.18	72.36	69.81	65.94
DSR [38]	non-uniform	93.75	93.86	93.13	72.31	71.98	70.70
RigL-ITOP [32]	uniform	93.19	92.08	89.36	70.46	68.39	64.16
RigL [11]	uniform	93.12	92.43	90.65	71.14	69.02	64.87
MEST+EM [54]	uniform	93.07	92.59	90.55	71.23	69.08	64.92
BiDST (ours)	uniform	**93.76±0.20**	**93.18±0.17**	**91.45±0.19**	**72.26±0.21**	**69.87±0.16**	**65.82 ±0.22**
RigL [11]	ERK	93.77	92.75	90.87	71.34	69.21	65.02
RigL-ITOP [32]	ERK	93.81	92.81	90.53	71.46	69.58	66.72
BiDST (ours)	ERK	**94.04±0.21**	**93.74±0.24**	**91.52±0.19**	**72.31±0.19**	**70.94±0.15**	**69.01 ±0.22**

Table B.2: MobileNet-v2 on CIFAR-10/100.

Method	Sparsity Distribution	Sparsity Ratio	Test Accuracy	Sparsity Ratio	Test Accuracy
		CIFAR-10		CIFAR-100	
Dense accuracy:		94.1%		73.5%	
BiDST	Uniform	0.9	91.8	0.95	71.0
	Uniform	0.95	91.1	0.95	69.8
	Uniform	0.98	89.7	0.98	68.3
BiDST	ERK	0.9	92.3	0.9	71.7
	ERK	0.95	91.7	0.95	70.6
	ERK	0.98	90.4	0.98	69.8

Table B.3: ResNet-34 on ImageNet.

Method	Sparsity Distribution	Sparsity Ratio	Test Accuracy
ImageNet-1K (*dense top-1 accuracy: 74.1%*)			
BiDST	Uniform	0.6	74.66
	Uniform	0.7	74.35
	Uniform	0.8	73.99
	Uniform	0.9	73.08
$\text{BiDST}_{5\times}$	Uniform	0.6	74.82
	Uniform	0.7	74.59
	Uniform	0.8	74.20
	Uniform	0.9	73.19



(a) Ablation study of S (b) Ablation study of λ

Figure C.1: Ablation study of the important hyperparameters of BiDST.