Advancing Model Pruning via Bi-level Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

The deployment constraints in practical applications necessitate the pruning of large-scale deep learning models, *i.e.*, promoting their weight sparsity. As illustrated by the Lottery Ticket Hypothesis (LTH), pruning also has the potential of improving their generalization ability. At the core of LTH, iterative magnitude pruning (IMP) is the predominant pruning method to successfully find 'winning tickets'. Yet, the computation cost of IMP grows prohibitively as the targeted pruning ratio increases. To reduce the computation overhead, various efficient 'one-shot' pruning methods have been developed but these schemes are usually unable to find winning tickets as good as IMP. This raises the question of *how to close the gap between pruning accuracy and pruning efficiency?* To tackle it, we pursue the algorithmic advancement of model pruning. Specifically, we formulate the pruning problem from a fresh and novel viewpoint, bi-level optimization (BLO). We show that the BLO interpretation provides a technically-grounded optimization base for an efficient implementation of the pruning-retraining learning paradigm used in IMP. We also show that the proposed bi-level optimization-oriented pruning method (termed BiP) is a special class of BLO problems with a bi-linear problem structure. By leveraging such bi-linearity, we theoretically show that BiP can be solved as easily as first-order optimization, thus inheriting the computation efficiency. Through extensive experiments on *both structured and unstructured pruning* with 5 model architectures and 4 data sets, we demonstrate that BiP can find better winning ticket than IMP in most cases, and is computationally as efficient as the one-shot pruning schemes, demonstrating 2-7$\times$ speedup over IMP for the same level of model accuracy and sparsity.

1 Introduction

While over-parameterized structures are key to the improved generalization of deep neural networks (DNNs) [1–3], they create new problems – the millions or even billions of parameters not only increase computational costs during inference, but also pose serious deployment challenges on resource-limited devices [4]. As a result, model pruning has seen a lot of research interest in recent years, focusing on reducing model sizes by removing (or pruning) redundant parameters [4–8]. Model sparsity (achieved by pruning) also benefits adversarial robustness [9], out-of-distribution generalization [10], and transfer learning [11]. Some pruning methods (towards structured sparsity) facilitate model deployment on hardware [12, 13].

Among various proposed model pruning algorithms [5, 9, 11, 14–27], the heuristics-based Iterative Magnitude Pruning (IMP) is the current dominant approach to achieving model sparsity without suffering performance loss, as suggested and empirically justified by the Lottery Ticket Hypothesis (LTH) [17]. The LTH hypothesizes the existence of a subnetwork (the so-called 'winning ticket') when trained in isolation (*e.g.*, from either a random initialization [17] or an early-rewinding point of dense model training [19]), can match the performance of the original dense model [5, 11, 17–20]. The core idea of IMP is to iteratively prune and retrain the model while progressively pruning

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

a small ratio of the remaining weights in each iteration, and continuing till the desired pruning ratio has been reached. While IMP often finds the winning ticket, it incurs the cost of repeated model retraining from scratch, making it prohibitively expensive for large datasets or large model architectures [28, 29]. To improve the efficiency of model pruning, numerous heuristics-based one-shot pruning methods [17, 21–25], *e.g.*, one-shot magnitude pruning (OMP), have been proposed. These schemes directly prune the model to the target sparsity and are significantly more efficient than IMP. Yet, the promise of current one-shot pruning methods is dataset/model-specific [22, 30] and mostly lies in the low pruning ratio regime [25]. As systematically studied in [22, 31], there exists a clear performance gap between one-shot pruning and IMP. As an alternative to heuristics-based schemes, optimization-based pruning methods [15, 16, 9, 26, 27] still follow the pruning-retraining paradigm and adopt sparsity regularization [32, 33] or parameterized masking [16, 9, 27] to prune models efficiently. However, these methods do not always match the accuracy of IMP and thus have not been widely used to find winning tickets [17, 21, 23–25, 29]. The empirical results showing that optimization underperforms heuristics motivate us to revisit the algorithmic fundamentals of pruning.

To this end, we put forth a novel perspective of model pruning as a bi-level optimization (BLO) problem. In this new formulation, we show that BLO provides a technically-grounded optimization basis for an efficient implementation of the pruning-retraining paradigm, the key algorithmic component used in IMP. To the best of our knowledge, we make the first rigorous connection between model pruning and BLO. Technically, we propose a novel **bi**-level optimization-enabled **p**runing method (termed BiP). We further show how BiP takes advantage of the bi-linearity of the pruning problem to avoid the computational challenges of common BLO methods, and is as efficient as any first-order alternating optimization scheme. Practically, we demonstrate the superiority of the proposed BiP in terms of accuracy, sparsity, and computation efficiency through extensive experiments. BiP finds the best winning ticket nearly in all settings while taking time comparable to the one-shot OMP. In Fig. 1, we present a snapshot of our empirical results for CIFAR-10 with 3 ResNet architectures at 74% pruning ratio. In all cases, BiP (⋆) finds winning tickets, improving accuracy over the dense model (•) and matching IMP (▲), while being upto 5× faster than IMP.

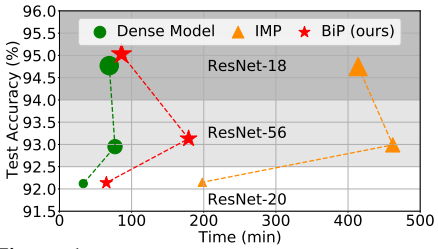


Figure 1: A performance snapshot of the proposed BiP method vs. the IMP baseline and the original dense model across three pruned ResNet models (ResNet-20, ResNet-56, and ResNet-18) with 74% sparsity on CIFAR-10. The marker size indicates the relative model size. The unicolor region corresponds to the same model type used by different pruning methods.

Our **contributions** can be summarized as follows:

• (Formulation) We rethink the algorithmic foundation of model pruning through the lens of BLO (bi-level optimization). The new BLO-oriented formulation disentangles pruning and retraining variables, providing the flexibility to design the interface between pruning and retraining.

• (Algorithm) We propose the new bi-level pruning (BiP) algorithm, which is built upon the aforementioned BLO formulation and the implicit gradient-based optimization theory. Unlike computationally intensive standard BLO solvers, we theoretically show that BiP is as efficient as any first-order optimization by taking advantage of the bi-linear nature of the pruning variables.

• (Experiments) We conduct extensive experiments across 4 datasets (CIFAR-10/100, Tiny-ImageNet and ImageNet), 5 model architectures, and 2 pruning settings (unstructured and structured pruning). We show that (i) BiP achieves higher test accuracy than IMP and finds the best winning tickets nearly in all settings, and (ii) BiP is highly efficient (comparable to one-shot pruning schemes), achieving 2-7× speedup over IMP for the same level of model accuracy and sparsity.

2 Related Work and Open Question

Neural network pruning. As neural networks become deeper and more sophisticated, model pruning technology has gained increasing attention over the last decade since pruned models are necessary for the deployment of deep networks in practical applications [4, 34]. With the goal of finding highly-sparse *and* highly-accurate subnetworks from original dense models, a variety of pruning methods have been developed such as heuristics-based pruning [17, 21, 23–25, 29, 35] and optimization-based pruning [16, 9, 26, 27]. The former identifies redundant model weights by leveraging heuristics-based metrics such as weight magnitudes [6, 17, 19, 11, 22, 36, 31, 35, 37], gradient magnitudes [21, 23, 24, 38, 39], and Hessian statistics [40–45].The latter is typically built on:

1) sparsity-promoting optimization [15, 33, 46–49], where model weights are trained by penalizing their sparsity-inducing norms, such as ℓ_0 and ℓ_1 norms for irregular weight pruning, and ℓ_2 norm for structured pruning; 2) parameterized masking [16, 9, 50–54], where model weight scores are optimized to filter the most important weights and achieve better performance.

Iterative vs. one-shot pruning, and motivation. Existing schemes can be further categorized into one-shot or iterative pruning based on the pruning schedule employed for achieving the targeted model sparsity. Among the iterative schemes, the IMP (Iterative Magnitude Pruning scheme) [17, 20, 55–64, 35] has played a significant role in identifying high-quality 'winning tickets', as postulated by LTH (Lottery Ticket Hypothesis) [18, 19]. To enable consistent comparisons among different methods, we extend the original definition of winning tickets in [17] to 'matching subnetworks' [20] so as to cover different implementations of winning tickets, *e.g.*, the use of early-epoch rewinding for model re-initialization [18] and the no-rewinding (*i.e.*, fine-tuning) variant [65]. Briefly, the matching subnetworks should match or surpass the performance of the original dense model [20].

For example, the current state-of-the-art (SOTA) implementation of IMP in [22] can lead to a pruned ResNet-20 on CIFAR-10 with 74% sparsity and 92.12% test accuracy, matching the performance of the original dense model (see red $\star$ in Fig. 2-**Left**). The IMP algorithm typically contains two key ingredients: **(i)** a *temporally-evolving pruning schedule* to progressively increase model sparsity over pruning iterations, and **(ii)** the *pruning-retraining learning mechanism* applied at each pruning iteration. With a target pruning ratio of $p\%$ with T pruning iterations, an example pruning schedule in (i) could be as follows – each iteration prunes $(p\%)^{1/T}$ of the currently unpruned model weights, progressively pruning fewer weights in each iteration. For (ii), the unpruned weights in each pruning iteration are re-set to the weights at initialization or at an early-training epoch [18], and re-trained till convergence. In brief, IMP repeatedly prunes, resets, and trains the network over multiple iterations.

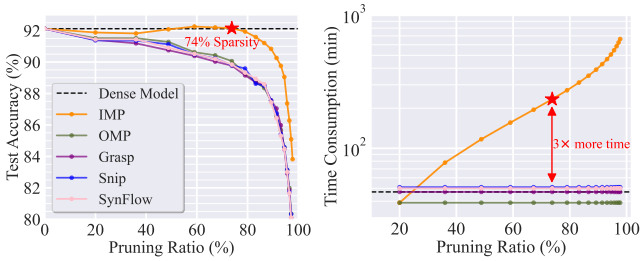


Figure 2: Illustration of the pros and cons of different pruning methods executed over (ResNet-20, CIFAR-10). **Left**: test accuracy vs. different pruning ratios of IMP and one-shot pruning methods (OMP [17], GRASP [23], SNIP [21], SYNFLOW [24]). **Right**: comparison of the efficiency with different sparsity.

However, winning tickets found by IMP incur significant computational costs. The sparsest winning ticket found by IMP in Fig. 2-**Left** (red $\star$) utilizes $T = 6$ pruning iterations. As shown in Fig. 2-**Right**, this takes $3\times$ more time than the original training of the dense model. To avoid the computational cost of IMP, different kinds of 'accelerated' pruning methods were developed [17, 21, 23–25, 29], and many fall into the *one-shot pruning* category: The network is directly pruned to the target sparsity and retrained once. In particular, OMP (one-shot magnitude pruning) is an important baseline that simplifies IMP [17]. It follows the pruning-retraining paradigm, but prunes the model to the target ratio with a single pruning iteration. Although one-shot pruning schemes are computationally cheap (Fig. 2-**Right**), they incur a significant accuracy drop compared to IMP (Fig. 2-**Left**). Even if IMP is customized with a reduced number of training epochs per pruning round, the pruning accuracy also drops largely (see Fig. A2). Hence, there is a need for advanced model pruning techniques to find winning tickets like IMP, while being efficient like one-shot pruning.

Different from unstructured weight pruning described above, structured pruning takes into consideration the sparse patterns of the model, such as filter and channel-wise pruning [47, 50, 66, 13, 67–70]. Structured pruning is desirable for deep model deployment in the presence of hardware constraints [4, 34]. However, compared to unstructured pruning, it is usually more challenging to maintain the performance and find structure-aware winning tickets [28, 29].

Open question. As discussed above, one-shot pruning methods are unable to match the predictive performance of IMP, and structure-aware winning tickets are hard to find. Clearly, the best optimization foundation of model pruning is underdeveloped. Thus, we ask:

> Is there an **optimization basis** for a successful pruning algorithm that can attain high pruned model accuracy (like IMP) *and* high computational efficiency (like one-shot pruning)?

The model pruning problem has a natural hierarchical structure – we need to find the best mask to prune model parameters, *and then*, given the mask, find the best model weights for the unpruned model parameters. Given this hierarchical structure, we believe that the *bi-level optimization (BLO) framework* is one promising optimization basis for a successful pruning algorithm.

Bi-level optimization (BLO). BLO is a general hierarchical optimization framework, where the upper-level problem depends on the solution to the lower-level problem. Such a nested structure makes the BLO in its most generic form very difficult to solve, since it is hard to characterize the influence of the lower-level optimization on the upper-level problem. Various existing literature focuses on the design and analysis of algorithms for various special cases of BLO. Applications range from the classical approximate descent methods [71, 72], penalty-based methods [73, 74], to recent BigSAM [75] and its extensions [76, 77]. It is also studied in the area of stochastic algorithms [78–80] and back-propagation based algorithms [81–83]. BLO has also advanced adversarial training [84], meta-learning [85], data poisoning attack generation [86], neural architecture search [87] as well as reinforcement learning [88]. Although BLO was referred in [49] for model pruning, it actually called an ordinary alternating optimization procedure, without taking the hierarchical learning structure of BLO into consideration. To the best of our knowledge, the BLO framework has not been considered for model pruning in-depth and systematically. We will show that model pruning yields a special class of BLO problems with bi-linear optimization variables. We will also theoretically show that this specialized BLO problem for model pruning can be solved as efficiently as first-order optimization. This is in a sharp contrast to existing BLO applications that rely on heuristics-based BLO solvers (*e.g.*, gradient unrolling in meta learning [85] and neural architecture search [87, 89]).

3 BiP: Model Pruning via Bi-level Optimization

In this section, we re-investigate the problem of model pruning through the lens of BLO and develop the bi-level pruning (BiP) algorithm. We can theoretically show that BiP can be solved as easily as the first-order alternating optimization by taking advantage of the bi-linearity of pruning variables.

A BLO viewpoint on model pruning As described in the previous section, the pruning-retraining learning paradigm covers two kinds of tasks: ❶ pruning that determines the sparse pattern of model weights, and ❷ training remaining non-zero weights to recover the model accuracy. In existing optimization-based pruning methods [90–93], the tasks ❶-❷ are typically achieved by optimizing model weights, together with penalizing their sparsity-inducing norms, *e.g.*, the ℓ_1 and ℓ_2 norms [94]. Different from the above formulation, we propose to separate optimization variables involved in the pruning tasks ❶ and ❷. This leads to the (binary) pruning mask variable $\mathbf{m} \in \{0,1\}^n$ and the model weight variable $\boldsymbol{\theta} \in \mathbb{R}^n$. Here n denotes the total number of model parameters. Accordingly, the pruned model is given by $(\mathbf{m} \odot \boldsymbol{\theta})$, where $\odot$ denotes the element-wise multiplication. As will be evident later, this form of *variable disentanglement* enables us to explicitly depict how the pruning and retraining process co-evolve, and helps customize the pruning task with high flexibility.

We next elaborate on how BLO can be established to co-optimize the pruning mask $\mathbf{m}$ and the retrained sparse model weights $\boldsymbol{\theta}$. Given the pruning ratio $p\%$, the sparsity constraint is given by $\mathbf{m} \in \mathcal{S}$, where $\mathcal{S} = \{\mathbf{m} \,|\, \mathbf{m} \in \{0,1\}^n, \mathbf{1}^T\mathbf{m} \leq k\}$ and $k = (1-p\%)n$. Our goal is to prune the original model *directly* to the targeted pruning ratio $p\%$ (*i.e.*, without calling for the IMP-like sparsity schedule as described in Sec. 2) and obtain the optimized sparse model $(\mathbf{m} \odot \boldsymbol{\theta})$. To this end, we interpret the pruning task (*i.e.*, ❶) and the model retraining task (*i.e.*, ❷) as *two optimization levels*, where the former is formulated as an *upper-level* optimization problem, and it relies on the optimization of the *lower-level* retraining task. We thus cast the model pruning problem as the following BLO problem (with ❷ being nested inside ❶):

$$\underset{\mathbf{m}\in\mathcal{S}}{\text{minimize}} \quad \underbrace{\ell(\mathbf{m} \odot \boldsymbol{\theta}^*(\mathbf{m}))}_{\text{❶: Pruning task}}; \qquad \text{subject to} \quad \underbrace{\boldsymbol{\theta}^*(\mathbf{m}) = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n} \overbrace{\ell(\mathbf{m} \odot \boldsymbol{\theta}) + \frac{\gamma}{2}\|\boldsymbol{\theta}\|_2^2}^{:= g(\mathbf{m},\boldsymbol{\theta})}}_{\text{❷: Sparsity-fixed model retraining}}, \tag{1}$$

where ℓ denotes the training loss (*e.g.*, cross-entropy), $\mathbf{m}$ and $\boldsymbol{\theta}$ are the upper-level and lower-level optimization variables respectively, $\boldsymbol{\theta}^*(\mathbf{m})$ signifies the lower-level solution obtained by minimizing the objective function g given the pruning mask $\mathbf{m}$, and $\gamma > 0$ is a regularization parameter introduced to convexify the lower-level optimization so as to stabilize the gradient flow from $\boldsymbol{\theta}^*(\mathbf{m})$ to $\mathbf{m}$ and thus the convergence of BLO [80, 95]. In a sharp contrast to existing single-level optimization-based model pruning methods [90–93], the BLO formulation (1) brings in two advantages.

First, BLO has the flexibility to use mismatched pruning and retraining objectives at the upper and lower optimization levels, respectively. This flexibility allows us to regularize the lower-level training

objective function in (1) and customize the implemented optimization methods at both levels. To be more specific, one can update the upper-level pruning mask $\mathbf{m}$ using a data batch (called $\mathcal{B}_2$) distinct from the one (called $\mathcal{B}_1$) used for obtaining the lower-level solution $\boldsymbol{\theta}^*(\mathbf{m})$. The resulting BLO procedure can then mimic the idea of meta learning to improve model generalization [96], where the lower-level problem fine-tunes $\boldsymbol{\theta}$ using $\mathcal{B}_1$, and the upper-level problem validates the generalization of the sparsity-aware finetuned model $(\mathbf{m} \odot \boldsymbol{\theta}^*(\mathbf{m}))$ using $\mathcal{B}_2$.

Second, BLO enables us to explicitly model and optimize the coupling between the retrained model weights $\boldsymbol{\theta}^*(\mathbf{m})$ and the pruning mask $\mathbf{m}$ through the implicit gradient (IG)-based optimization routine. Here IG refers to the gradient of the lower-level solution $\boldsymbol{\theta}^*(\mathbf{m})$ with respect to (w.r.t.) the upper-level variable $\mathbf{m}$, and its derivation calls the implicit function theory [74]. The use of IG makes our proposed BLO-oriented pruning (1) significantly different from the greedy alternating minimization [97] that learns the upper-level and lower-level variables independently (*i.e.*, minimizes one variable by fixing the other). We refer readers to the following section for the detailed IG theory. We will also show in Sec. 4 that the pruning strategy from (1) can outperform IMP in many pruning scenarios but is much more efficient as it does not call for the scheduler of iterative pruning ratios.

Optimization foundation of BIP. The key optimization challenge of solving the BIP problem (1) lies in the computation of IG (implicit gradient). Prior to developing an effective solution, we first elaborate on the *IG challenge*, the unique characteristic of BLO. In the context of gradient descent, the gradient of the objective function in (1) yields

$$\underbrace{\frac{d\ell(\mathbf{m} \odot \boldsymbol{\theta}^*(\mathbf{m}))}{d\mathbf{m}}}_{\text{Gradient of objective}} = \nabla_{\mathbf{m}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*(\mathbf{m})) + \underbrace{\frac{d(\boldsymbol{\theta}^*(\mathbf{m})^\top)}{d\mathbf{m}}}_{\text{IG}} \nabla_{\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*(\mathbf{m})), \tag{2}$$

where $\nabla_{\mathbf{m}}$ and $\nabla_{\boldsymbol{\theta}}$ denote the *partial derivatives* of the bi-variate function $\ell(\mathbf{m} \odot \boldsymbol{\theta})$ w.r.t. the variable $\mathbf{m}$ and $\boldsymbol{\theta}$ respectively, $d\boldsymbol{\theta}^\top / d\mathbf{m} \in \mathbb{R}^{n \times n}$ denotes the vector-wise *full derivative*, and for ease of notation, we will omit the transpose $\top$ when the context is clear. In (2), the IG challenge refers to the demand for computing the full gradient of the implicit function $\boldsymbol{\theta}^*(\mathbf{m}) = \arg\min_{\boldsymbol{\theta}} g(\mathbf{m}, \boldsymbol{\theta})$ w.r.t. $\mathbf{m}$, where recall from (1) that $g(\mathbf{m}, \boldsymbol{\theta}) := \ell(\mathbf{m} \odot \boldsymbol{\theta}) + \frac{\gamma}{2}\|\boldsymbol{\theta}\|_2^2$.

Next, we derive the IG formula following the rigorous implicit function theory [74, 80, 85]. Based on the fact that $\boldsymbol{\theta}^*(\mathbf{m})$ satisfies the stationarity condition for the lower-level objective function in (2), it is not difficult to obtain that (see derivation in Appendix A)

$$\frac{d\boldsymbol{\theta}^*(\mathbf{m})}{d\mathbf{m}} = -\nabla^2_{\mathbf{m}\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*)[\nabla^2_{\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*) + \gamma \mathbf{I}]^{-1}, \tag{3}$$

where $\nabla^2_{\mathbf{m}\boldsymbol{\theta}} \ell$ and $\nabla^2_{\boldsymbol{\theta}} \ell$ denote the second-order partial derivatives of ℓ respectively, and $(\cdot)^{-1}$ denotes the matrix inversion operation.

Yet, the exact IG formula (3) remains difficult to calculate due to the presence of matrix inversion and second-order partial derivatives. To simplify it, we impose the Hessian-free assumption, $\nabla^2_{\boldsymbol{\theta}} \ell = \mathbf{0}$, which is mild in general; For example, the decision boundaries of neural networks with ReLU activations are piece-wise linear in a tropical hyper-surface [98], and this assumption has been widely used in BLO-involved applications such as meta learning [99] and adversarial learning [84]. Given $\nabla^2_{\boldsymbol{\theta}} \ell = \mathbf{0}$, the matrix inversion in (3) can be then mitigated, leading to the IG formula

$$\frac{d\boldsymbol{\theta}^*(\mathbf{m})}{d\mathbf{m}} = -\frac{1}{\gamma} \nabla^2_{\mathbf{m}\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*). \tag{4}$$

At the first glance, the computation of the simplified IG (4) still requires the mixed (second-order) partial derivative $\nabla^2_{\mathbf{m}\boldsymbol{\theta}} \ell$. However, BIP is a special class of BLO problems with bi-linear variables $(\mathbf{m} \odot \boldsymbol{\theta})$. Based on this bi-linearity, we can prove that IG in (4) can be *analytically* expressed using only *first-order* derivatives; see the following theorem.

Proposition 1 *Assuming $\nabla^2_{\boldsymbol{\theta}} \ell = 0$ and defining $\nabla_{\mathbf{z}} \ell(\mathbf{z}) := \nabla_{\mathbf{z}} \ell(\mathbf{z}) \,|_{\mathbf{z} = \mathbf{m} \odot \boldsymbol{\theta}^*}$, the implicit gradient (4) is then given by*

$$\frac{d\boldsymbol{\theta}^*(\mathbf{m})}{d\mathbf{m}} = -\frac{1}{\gamma} \operatorname{diag}(\nabla_{\mathbf{z}} \ell(\mathbf{z})); \tag{5}$$

Further, the gradient of the objective function given by (2) becomes

$$\frac{d\ell(\mathbf{m} \odot \boldsymbol{\theta}^*)}{d\mathbf{m}} = (\boldsymbol{\theta}^* - \frac{1}{\gamma} \mathbf{m} \odot \nabla_{\mathbf{z}} \ell(\mathbf{z})) \odot \nabla_{\mathbf{z}} \ell(\mathbf{z}), \tag{6}$$

where $\odot$ denotes the element-wise multiplication.

Proof: Using chain-rule, we can obtain that

$$\nabla_{\boldsymbol{\theta}}\ell(\mathbf{m}\odot\boldsymbol{\theta}^*) = \mathrm{diag}(\mathbf{m})\nabla_{\mathbf{z}}\ell(\mathbf{z}) = \mathbf{m}\odot\nabla_{\mathbf{z}}\ell(\mathbf{z}); \tag{7}$$

$$\text{similarly, } \nabla_{\mathbf{m}}\ell(\mathbf{m}\odot\boldsymbol{\theta}^*) = \mathrm{diag}(\boldsymbol{\theta}^*)\nabla_{\mathbf{z}}\ell(\mathbf{z}) = \boldsymbol{\theta}^*\odot\nabla_{\mathbf{z}}\ell(\mathbf{z}) \tag{8}$$

where $\mathrm{diag}(\cdot)$ represents a diagonal matrix with $\cdot$ being the main diagonal vector. Further, we can convert (4) to

$$\begin{aligned}\nabla^2_{\mathbf{m}\boldsymbol{\theta}}\ell(\mathbf{m}\odot\boldsymbol{\theta}^*) &\overset{(7)}{=} \nabla_{\mathbf{m}}\left[\mathbf{m}\odot\nabla_{\mathbf{z}}\ell(\mathbf{z})\right] \overset{\text{chain rule}}{=} \mathrm{diag}(\nabla_{\mathbf{z}}\ell(\mathbf{z})) + \mathrm{diag}(\mathbf{m})[\nabla_{\mathbf{m}}(\nabla_{\mathbf{z}}\ell(\mathbf{z}))] \\ &\overset{(8)}{=} \mathrm{diag}(\nabla_{\mathbf{z}}\ell(\mathbf{z})) + \mathrm{diag}(\mathbf{m})[\mathrm{diag}(\boldsymbol{\theta}^*)\nabla^2_{\mathbf{z}}\ell(\mathbf{z})] = \mathrm{diag}(\nabla_{\mathbf{z}}\ell(\mathbf{z})),\end{aligned} \tag{9}$$

where the last equality holds due to the Hessian-free assumption. With (9) and (4) we can prove (5).

Next, substituting the IG (5) to the upper-level gradient (2), we obtain that

$$\begin{aligned}\frac{d\ell(\mathbf{m}\odot\boldsymbol{\theta}^*)}{d\mathbf{m}} &= \nabla_{\mathbf{m}}\ell(\mathbf{m}\odot\boldsymbol{\theta}^*) - \frac{1}{\gamma}\nabla_{\mathbf{z}}\ell(\mathbf{z})\odot\nabla_{\boldsymbol{\theta}}\ell(\mathbf{m}\odot\boldsymbol{\theta}^*) \\ &\overset{(7),(8)}{=} \boldsymbol{\theta}^*\odot\nabla_{\mathbf{z}}\ell(\mathbf{z}) - \frac{1}{\gamma}\nabla_{\mathbf{z}}\ell(\mathbf{z})\odot(\mathbf{m}\odot\nabla_{\mathbf{z}}\ell(\mathbf{z})) = (\boldsymbol{\theta}^* - \frac{1}{\gamma}\mathbf{m}\odot\nabla_{\mathbf{z}}\ell(\mathbf{z}))\odot\nabla_{\mathbf{z}}\ell(\mathbf{z}),\end{aligned}$$

which leads to (6). The proof is now complete. □

The key insight drawn from Prop. 1 is that the bi-linearity of pruning variables (*i.e.*, $\mathbf{m}\odot\boldsymbol{\theta}^*$) makes the IG-involved gradient (2) easily solvable, and the computational complexity is almost the same as that of computing the first-order gradient $\nabla_{\mathbf{z}}\ell(\mathbf{z})$ just once, as supported by (6)

BIP algorithm and implementation. We next formalize the BIP algorithm based on Prop. 1 and the alternating gradient descent based BLO solver [80]. At iteration t, there are two main steps.

★ *Lower-level SGD for model retraining*: Given $\mathbf{m}^{(t-1)}$, $\boldsymbol{\theta}^{(t-1)}$, and $\mathbf{z}^{(t-1)} := \mathbf{m}^{(t-1)}\odot\boldsymbol{\theta}^{(t-1)}$, we update $\boldsymbol{\theta}^{(t)}$ by randomly selecting a data batch with the learning rate α and applying SGD (stochastic gradient descent) to the lower-level problem of (1),

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \alpha\nabla_{\boldsymbol{\theta}}g(\mathbf{m}^{(t-1)},\boldsymbol{\theta}^{(t-1)}) \overset{(7)}{=} \boldsymbol{\theta}^{(t-1)} - \alpha[\mathbf{m}^{(t-1)}\odot\nabla_{\mathbf{z}}\ell(\mathbf{z})\,|_{\mathbf{z}=\mathbf{z}^{(t-1)}} + \gamma\boldsymbol{\theta}^{(t-1)}], \quad (\boldsymbol{\theta}\text{-step})$$

★ *Upper-level SPGD for pruning*: Given $\mathbf{m}^{(t-1)}$, $\boldsymbol{\theta}^{(t)}$, and $\mathbf{z}^{(t+1/2)} := \mathbf{m}^{(t-1)}\odot\boldsymbol{\theta}^{(t)}$, we update $\mathbf{m}$ using SPGD (stochastic projected gradient descent) along the IG-enhanced descent direction (2),

$$\begin{aligned}\mathbf{m}^{(t)} &= \mathcal{P}_{\mathcal{S}}\left[\mathbf{m}^{(t-1)} - \beta\frac{d\ell(\mathbf{m}\odot\boldsymbol{\theta}^{(t)})}{d\mathbf{m}}\,|_{\mathbf{m}=\mathbf{m}^{(t-1)}}\right] \\ &\overset{(6)}{=} \mathcal{P}_{\mathcal{S}}\left[\mathbf{m}^{(t-1)} - \beta\left(\boldsymbol{\theta}^{(t)} - \frac{1}{\gamma}\mathbf{m}^{(t-1)}\odot\nabla_{\mathbf{z}}\ell(\mathbf{z})\,|_{\mathbf{z}=\mathbf{z}^{(t+1/2)}}\right)\odot\nabla_{\mathbf{z}}\ell(\mathbf{z})\,|_{\mathbf{z}=\mathbf{z}^{(t+1/2)}}\right], \quad (\mathbf{m}\text{-step})\end{aligned}$$

where $\beta > 0$ is the upper-level learning rate, and $\mathcal{P}_{\mathcal{S}}(\cdot)$ denotes the Euclidean projection onto the constraint set $\mathcal{S}$ given by $\mathcal{S} = \{\mathbf{m}\,|\,\mathbf{m}\in\{0,1\}^n, \mathbf{1}^T\mathbf{m}\le k\}$ in (1) and is achieved by the top-k hard-thresholding operation as will be detailed later.

In BIP, the ($\boldsymbol{\theta}$-step) and ($\mathbf{m}$-step) steps execute iteratively. In practice, we find that BIP converges very well; see Fig. A8 for an ablation study. It is also worth noting that both ($\boldsymbol{\theta}$-step) and ($\mathbf{m}$-step) only require the first-order information $\nabla_{\mathbf{z}}\ell(\mathbf{z})$, demonstrating that BIP can be conducted as efficiently as first-order optimization. We present more implementation details of BIP below and refer readers to Appendix B for a detailed algorithm description.

✦ *Discrete optimization over* $\mathbf{m}$*:* We follow the 'convex relaxation + hard thresholding' mechanism used in [16, 9]. Specifically, we relax the binary masking variables to continuous masking scores $\mathbf{m}\in[\mathbf{0},\mathbf{1}]$. We then acquire loss gradients at the backward pass based on the relaxed $\mathbf{m}$. At the forward pass, we project it onto the discrete constraint set $\mathcal{S}$ using the hard thresholding operator, where the top k elements are set to 1s and the others to 0s. See Appendix B for more discussion.

✦ *Data batch selection for lower-level and upper-level optimization:* We adopt different data batches (with the same batch size) when implementing ($\boldsymbol{\theta}$-step) and ($\mathbf{m}$-step). This is one of the advantages of the BLO formulation, which enables the flexibility to customize the lower-level and upper-level problems. The use of diverse data batches is beneficial to generalization as shown in [96].

✦ *Hyperparameter tuning:* As described in ($\boldsymbol{\theta}$-step)-($\mathbf{m}$-step), BIP needs to set two learning rates α and β for lower-level and upper-level optimization, respectively. We choose $\alpha = 0.01$ and $\beta = 0.1$ in

all experiments, where we adopt the mask learning rate β from Hydra [9] and set a smaller lower-level learning rate α, as $\boldsymbol{\theta}$ is initialized by a pre-trained dense model. We show ablation study on α in Fig. 6(c). BLO also brings in the low-level convexification parameter γ. We set $\gamma = 1.0$ in experiments and refer readers to Fig. 6(b) for a sanity check.

✦ *One-step vs. multi-step SGD:* In ($\boldsymbol{\theta}$-step), the one-step SGD is used and helps reduce the computation overhead. In practice, we also find that the one-step SGD is sufficient: The use of multi-step SGD in BiP does not yield much significant improvement over the one-step version; see Fig. 6(a).

✦ *Extension to structured pruning:* We formulate and solve the BiP problem in the context of unstructured (element-wise) weight pruning. However, if define the pruning mask $\mathbf{m}$ *w.r.t.* model's structural units (*e.g.*, filters), then it is easy to apply BiP to structured weight pruning; see Fig. 4 and Fig. A6 for empirical results.

4 Experiments

In this section, we present extensive experimental results to show the effectiveness of BiP across multiple model architectures, various datasets, and different pruning setups. Compared to IMP, one-shot pruning, and optimization-based pruning baselines, we find that BiP can find better winning tickets in most cases and is computationally efficient.

4.1 Experiment Setup

Datasets and models. Following the pruning benchmark in [22], we consider 4 datasets including CIFAR-10 [100], CIFAR-100 [100], Tiny-ImageNet [101], ImageNet [102], and 5 architecture types including ResNet-20/56/18/50 and VGG-16 [103, 104]. Tab. A1 summarizes these datasets and model configurations and setups.

Baselines, training, and evaluation. As baselines, we mainly focus on 4 SOTA pruning methods, ① IMP [17], ② OMP [17], ③ GRASP [23] (a one-shot pruning method by analyzing gradient flow at initialization), and ④ HYDRA [9] (an optimization-based pruning method that optimizes masking scores). It is worth noting that there exist various implementations of IMP, *e.g.*, specified by different learning rates and model initialization or 'rewinding' strategies [18]. To make a fair comparison, we follow the recent IMP benchmark in [22], which can find the best winning tickets over current heuristics-based pruning baselines. We also remark that HYDRA is originally proposed for improving the adversarial robustness of a pruned model, but it can be easily customized for standard pruning when setting the adversary's strength as 0 [9]. We choose HYDRA as a baseline because it can be regarded as a single-level variant of BiP with post-optimization weight retraining. When implementing BiP, unless specified otherwise, we use the 1-step SGD in ($\boldsymbol{\theta}$-step), and set the learning rates (α, β) and the lower-level regularization parameter γ as described in the previous section. When implementing baselines, we follow their official repository setups. We evaluate the performance of all methods mainly from two perspectives: (1) the test accuracy of the sub-network, and (2) the runtime of pruning to reach the desired sparsity. We refer readers to Tab. A3 and Appendix C.2 for more training and evaluation details, such as training epochs and learning rate schedules.

4.2 Experiment Results

BiP identifies high-accuracy winning tickets. In what follows, we look at the quality of winning tickets identified by BiP. *Two key observations* can be drawn from our results: (1) BiP finds winning tickets of higher accuracy and/or higher sparsity than the baselines in most cases (as shown in Fig. 3 and Tab. 1); (2) The superiority of BiP holds for both unstructured pruning and structured pruning (as shown in Fig. 4 and Fig. A6).

Fig. 3 shows the *unstructured pruning trajectory* (given by test accuracy vs. pruning ratio) of BiP and baseline methods in 8 model-dataset setups. For comparison, we also present the performance of the original dense model. As we can see, the proposed BiP approach finds the best winning tickets (in terms of the highest accuracy) compared to the baselines across all the pruning setups. Among the baseline methods, IMP is the pruning method most competitive to ours. However, the improvement brought by BiP is significant with respect to the variance of IMP, except for the 60%-80% sparsity regime in (CIFAR-10, ResNet-20). In the case of (CIFAR-100, ResNet-20) where IMP can not find any winning tickets (as confirmed by [22]), BiP still manages to find winning tickets with around 0.6% improvement over the dense model. In Tab. 1, we summarize the sparsest winning tickets along the pruning trajectory identified by different pruning methods. As we can see, BiP can identify the winning tickets with higher sparsity levels than the other methods, except in the case of (CIFAR-10,

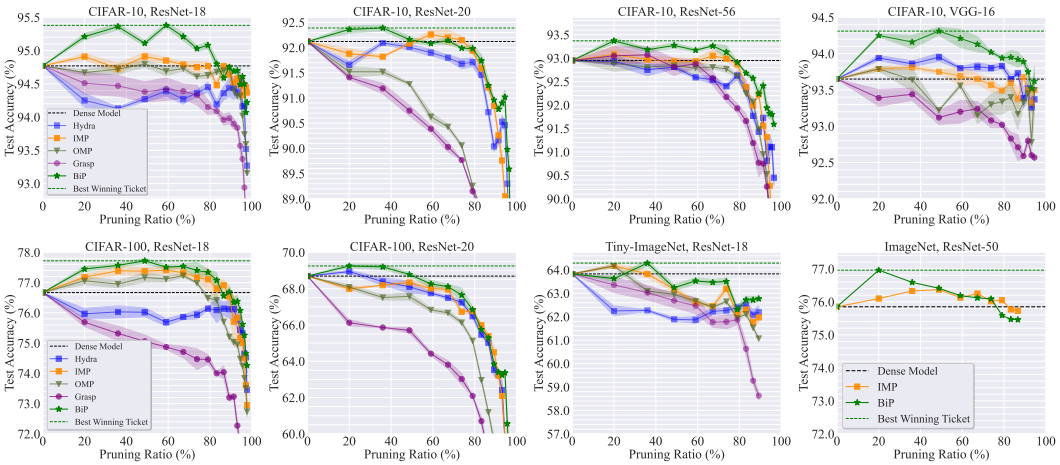


Figure 3: Unstructured pruning trajectory given by test accuracy (%) vs. sparsity (%) on various dataset-model pairs. The proposed BIP is compared with HYDRA [9], IMP [22], OMP [22], GRASP [23]. And the performance of dense model and that of the best winning ticket are marked using dashed lines in each plot. The solid line and shaded area of each pruning method represent the mean and variance of test accuracies over 3 independent trials. We observe that BIP consistently outperforms the other baselines. Note in the (ImageNet, ResNet-50) setting, we only compare BIP with our strongest baseline IMP due to computational resource constraints.

ResNet-20). We also refer readers to Appendix C.3 for more experimental results to justify the high quality of winning tickets identified by BIP.

Table 1: The sparsest winning tickets found by different methods at different data-model setups. winning tickets refer to the sparse models with an average test accuracy no less than the dense model [20]. In each cell, $p\%$ (acc$\pm$std%) represents the sparsity as well as the test accuracy. The test accuracy of dense models can be found in the header. ✗ signifies that no winning ticket is found by a pruning method. Given the data-model setup (*i.e.*, per column), the sparsest winning ticket is highlighted in **bold**.

Method	CIFAR-10				CIFAR-100	
	ResNet-18 (94.77%)	ResNet-20 (92.12%)	ResNet-56 (92.95%)	VGG-16 (93.65%)	ResNet-18 (76.67%)	ResNet-20 (68.69%)
IMP	87% (94.77±0.10%)	**74**% (92.15±0.15%)	**74**% (92.99±0.12%)	89% (93.68±0.05%)	87% (76.91±0.19%)	✗
OMP	49% (94.80±0.10%)	✗	✗	20% (93.79±0.06%)	74% (76.99±0.07%)	✗
GRASP	✗	✗	36% (93.07±0.34%)	✗	✗	✗
HYDRA	✗	✗	✗	87% (93.73±0.03%)	✗	20% (68.94±0.17%)
BIP	**89**% (94.79±0.15%)	67% (92.14±0.15%)	**74**% (93.13±0.04%)	**93**% (93.75±0.15%)	**89**% (76.69±0.18%)	**49**% (68.78±0.10%)

Fig. 4 demonstrates the *structured pruning trajectory* on the CIFAR-10/100 datasets. Here we focus on filter pruning, where the filter is regarded as a masking unit in (1). We refer readers to Appendix C.3 for channel-wise pruning results. Compared to Fig. 3, Fig. 4 shows that it becomes more difficult to find winning tickets of high accuracy and sparsity in the structured pruning, and the gap among different methods decreases. This is not surprising, since filter pruning imposes much stricter pruning structure constraints than irregular pruning. However, BIP still outperforms all the baselines. Most importantly, it identifies clear winning tickets in the low sparse regime even when IMP fails.

BIP is computationally efficient. In our experiments, another **key observation** is that BIP yields sparsity-agnostic runtime complexity while IMP leads to runtime exponential to the target sparsity. Fig. 5 shows the computation cost of different methods versus pruning ratios on (CIFAR-10, ResNet-18). For example, BIP takes 86 mins to find the sparsest winning ticket (with 89% sparsity in Tab. 1).

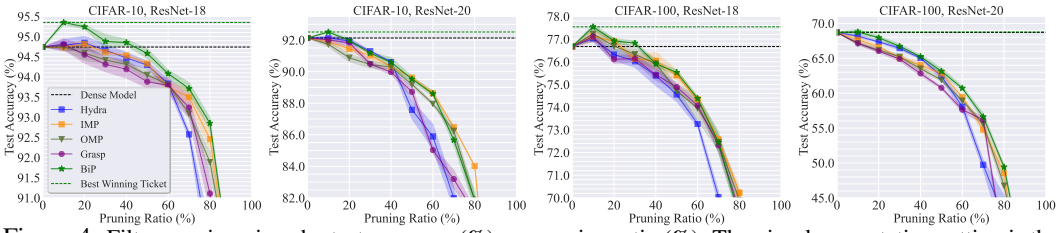


Figure 4: Filter pruning given by test accuracy (%) vs. pruning ratio (%). The visual presentation setting is the same as Fig. 3. We observe that BIP identifies winning tickets of structured pruning in certain sparsity regimes.

This yields $7\times$ less runtime than IMP, which consumes 620 mins to find a comparable winning ticket with 87% sparsity. Compared to the optimization-based baseline HYDRA, BIP is more efficient as it does not rely on the extra post-optimization retraining; see Tab. A2 for a detailed summary of runtime and number of training epochs required by different pruning methods. Further, BIP takes about 1.25 $\times$ more computation time than GRASP and OMP. However, the latter methods lead to worse pruned model accuracy, as demonstrated by their failure to find winning tickets in Tab. 1, Fig. 3, and Fig. 4.

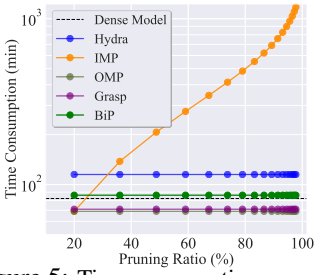


Figure 5: Time consumption comparison on (CIFAR-10, ResNet-18) with different pruning ratio p.

Rewinding effect on BIP. Another advantage of BIP is that it is *not* sensitive to model rewinding. Recall that rewinding is a strategy used in LTH [19] to determine what model initialization should be used for re-training a pruned sparse model. As shown in [22], an IMP-identified winning ticket could be sensitive to the choice of a rewinding point. Fig. 6(a) shows the test accuracy of the BIP-pruned model when it is retrained at different rewinding epochs, where 'N/A' in the x-axis represents the case of no retraining (and thus no rewinding). More ablation studies on the rewinding epochs can be found in Fig. A11. As we can see, a carefully-tuned rewinding scheme does not lead to a significant improvement over BIP without retraining. This suggests that the subnetworks found by BIP are already of high quality.

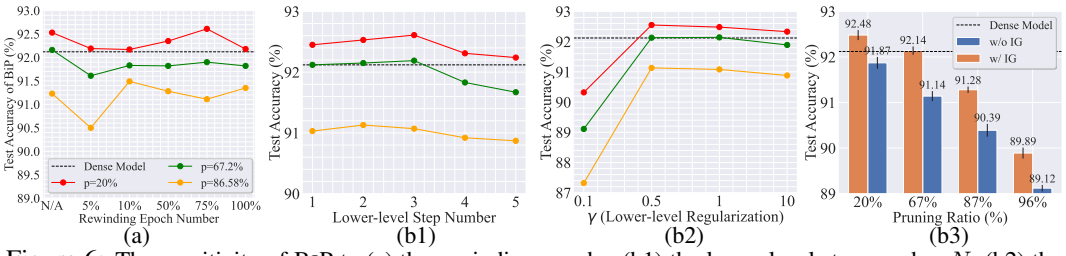


Figure 6: The sensitivity of BIP to (a) the rewinding epochs, (b1) the lower-level step number N, (b2) the lower-level regularizer γ, and (b3) the contribution of the IG-term at various pruning ratios on (CIFAR-10, ResNet-20). Each curve or column represents a certain sparsity. In (a), "N/A" in the x-axis indicates BIP without retraining. In (b3), we fix the γ to 1.0 and compare the performance of the IG-involved/excluded (2) BIP.

Sanity check of BIP on specialized hyperparameters. In Fig.6(b), we show the sensitivity of BIP to its specialized hyperparameters at lower-level optimization, including the number of SGD iterations (N) in ($\boldsymbol{\theta}$-step), and the regularization parameter γ in (1). Fig. 6(b1) shows the test accuracy of BIP-pruned models versus the choice of N. As we can see, more SGD iterations for the lower-level optimization do not improve the performance of BIP. This is because in BIP, the $\boldsymbol{\theta}$-step is initialized by a pre-trained model which does not ask for aggressive weight updating. The best performance of BIP is achieved at $N \leq 3$. We choose $N = 1$ throughout the experiments as it is computationally lightest. Fig. 6(b2) shows the performance of BIP by varying γ. As we can see, the choice of $\gamma \in \{0.5, 1\}$ yields the best pruning accuracy across all pruning ratios. If γ is too small, the lack of strong convexity of lower-level optimization would hamper the convergence. If γ is too large, the lower-level optimization would depart from the true model objective and causes a performance drop. Fig. 6(b3) demonstrates the necessity of the IG enhancement in BIP. We compare BIP with its IG-free version by dropping the IG term in (2). We observe that BIP without IG (marked in blue) leads to a significant performance drop ($> 1\%$) at various sparsities. This highlights that *the IG in the* ($\mathbf{m}$*-step) plays a critical role in the performance improvements obtained by* BIP, justifying our novel BLO formulation for the pruning problem. See Fig. A4 for more results on hyperparameter tuning.

5 Conclusion

We proposed the BIP method to find sparse networks through the lens of BLO. Our work advanced the algorithmic foundation of model pruning by characterizing its pruning-retraining hierarchy using BLO. We theoretically showed that BIP can be solved as easily as first-order optimization by exploiting the bi-linearity of pruning variables. We also empirically showed that BIP can find high-quality winning tickets very efficiently compared to the predominant iterative pruning method. In the future, we will seek the optimal curriculum of training data at different optimization levels of BIP, and will investigate the performance of our proposal for actual hardware acceleration.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin, "Deep learning: a statistical viewpoint," *Acta numerica*, vol. 30, pp. 87–201, 2021.

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

[5] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag, "What is the state of neural network pruning?," *arXiv preprint arXiv:2003.03033*, 2020.

[6] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[7] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 784–800.

[8] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J Dally, "Exploring the granularity of sparsity in convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 13–20.

[9] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana, "Hydra: Pruning adversarially robust neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19655–19666, 2020.

[10] James Diffenderfer, Brian Bartoldson, Shreya Chaganti, Jize Zhang, and Bhavya Kailkhura, "A winning hand: Compressing deep networks can improve out-of-distribution robustness," *Advances in Neural Information Processing Systems*, vol. 34, pp. 664–676, 2021.

[11] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang, "The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16306–16316.

[12] Xiaolong Ma, Fu-Ming Guo, Wei Niu, Xue Lin, Jian Tang, Kaisheng Ma, Bin Ren, and Yanzhi Wang, "Pconv: The missing but desirable sparsity in dnn weight pruning for real-time execution on mobile devices," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 5117–5124.

[13] Tianlong Chen, Xuxi Chen, Xiaolong Ma, Yanzhi Wang, and Zhangyang Wang, "Coarsening the granularity: Towards structurally sparse lottery tickets," *arXiv preprint arXiv:2202.04736*, 2022.

[14] Yann LeCun, John Denker, and Sara Solla, "Optimal brain damage," *Advances in neural information processing systems*, vol. 2, 1989.

[15] Ao Ren, Tianyun Zhang, Shaokai Ye, Jiayu Li, Wenyao Xu, Xuehai Qian, Xue Lin, and Yanzhi Wang, "Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction method of multipliers," 2018.

[16] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari, "What's hidden in a randomly weighted neural network?," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11893–11902.

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

[18] Alex Renda, Jonathan Frankle, and Michael Carbin, "Comparing rewinding and fine-tuning in neural network pruning," in *8th International Conference on Learning Representations*, 2020.

[19] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin, "Linear mode connectivity and the lottery ticket hypothesis," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3259–3269.

[20] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin, "The lottery ticket hypothesis for pre-trained bert networks," *Advances in neural information processing systems*, vol. 33, pp. 15834–15846, 2020.

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

[22] Xiaolong Ma, Geng Yuan, Xuan Shen, Tianlong Chen, Xuxi Chen, Xiaohan Chen, Ning Liu, Minghai Qin, Sijia Liu, Zhangyang Wang, et al., "Sanity checks for lottery tickets: Does your winning ticket really win the jackpot?," *arXiv preprint arXiv:2107.00166*, 2021.

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

[24] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6377–6389, 2020.

[25] Milad Alizadeh, Shyam A. Tailor, Luisa M Zintgraf, Joost van Amersfoort, Sebastian Farquhar, Nicholas Donald Lane, and Yarin Gal, "Prospect pruning: Finding trainable weights at initialization using meta-gradients," 2022.

[26] Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin, "Layer-adaptive sparsity for the magnitude-based pruning," *arXiv preprint arXiv:2010.07611*, 2020.

[27] Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber, "Are neural nets modular? inspecting functional modularity through differentiable weight masks," *arXiv preprint arXiv:2010.02066*, 2020.

[28] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin, "Drawing early-bird tickets: Towards more efficient training of deep networks," *arXiv preprint arXiv:1909.11957*, 2019.

[29] Zhenyu Zhang, Xuxi Chen, Tianlong Chen, and Zhangyang Wang, "Efficient lottery ticket finding: Less data is more," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12380–12390.

[30] Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D Lee, "Sanity-checking pruning methods: Random tickets can win the jackpot," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20390–20401, 2020.

[31] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin, "Pruning neural networks at initialization: Why are we missing the mark?," *arXiv preprint arXiv:2009.08576*, 2020.

[32] Sijia Liu, Parikshit Ram, Deepak Vijaykeerthy, Djallel Bouneffouf, Gregory Bramble, Horst Samulowitz, Dakuo Wang, Andrew Conn, and Alexander Gray, "An ADMM based framework for automl pipeline configuration," 2019.

[33] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu, "Neural pruning via growing regularization," *arXiv preprint arXiv:2012.09243*, 2020.

[34] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas, "Predicting parameters in deep learning," *Advances in neural information processing systems*, vol. 26, 2013.

[35] Michael Zhu and Suyog Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[36] Steven A Janowsky, "Pruning versus clipping in neural networks," *Physical Review A*, vol. 39, no. 12, pp. 6600, 1989.

[37] Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh, "Ac/dc: Alternating compressed/decompressed training of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8557–8570, 2021.

[38] Michael C Mozer and Paul Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in neural information processing systems*, 1989, pp. 107–115.

[39] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen, "Rigging the lottery: Making all tickets winners," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2943–2952.

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

[41] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney, "Pyhessian: Neural networks through the lens of the hessian," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 581–590.

[42] Yann LeCun, John S Denker, and Sara A Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[43] Babak Hassibi and David G Stork, *Second order derivatives for network pruning: Optimal brain surgeon*, Morgan Kaufmann, 1993.

[44] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[45] Sidak Pal Singh and Dan Alistarh, "Woodfisher: Efficient second-order approximation for neural network compression," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18098–18109, 2020.

[46] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.

[47] Yihui He, Xiangyu Zhang, and Jian Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1389–1397.

[48] Hao Zhou, Jose M Alvarez, and Fatih Porikli, "Less is more: Towards compact cnns," in *European Conference on Computer Vision*. Springer, 2016, pp. 662–677.

[49] Christos Louizos, Max Welling, and Diederik P Kingma, "Learning sparse neural networks through l_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.

[50] Yi Guo, Huan Yuan, Jianchao Tan, Zhangyang Wang, Sen Yang, and Ji Liu, "Gdp: Stabilized neural network pruning via gates with differentiable polarization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5239–5250.

[51] Edgar Liberis and Nicholas D Lane, "Differentiable network pruning for microcontrollers," *arXiv preprint arXiv:2110.08350*, 2021.

[52] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi, "Soft threshold weight reparameterization for learnable sparsity," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5544–5555.

[53] Chao Xue, Xiaoxing Wang, Junchi Yan, Yonggang Hu, Xiaokang Yang, and Kewei Sun, "Rethinking bi-level optimization in neural architecture search: A gibbs sampling perspective," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 10551–10559.

[54] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang, "Effective sparsification of neural networks with global sparsity constraint," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3599–3608.

[55] Trevor Gale, Erich Elsen, and Sara Hooker, "The state of sparsity in deep neural networks," *arXiv*, vol. abs/1902.09574, 2019.

[56] Zhenyu Zhang, Xuxi Chen, Tianlong Chen, and Zhangyang Wang, "Efficient lottery ticket finding: Less data is more," in *Proceedings of the 38th International Conference on Machine Learning*, Marina Meila and Tong Zhang, Eds. 18–24 Jul 2021, vol. 139 of *Proceedings of Machine Learning Research*, pp. 12380–12390, PMLR.

[57] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang, "The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models," *arXiv preprint arXiv:2012.06908*, 2020.

[58] Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos, "Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp," in *8th International Conference on Learning Representations*, 2020.

[59] Xuxi Chen, Zhenyu Zhang, Yongduo Sui, and Tianlong Chen, "{GAN}s can play lottery tickets too," in *International Conference on Learning Representations*, 2021.

[60] Haoyu Ma, Tianlong Chen, Ting-Kuei Hu, Chenyu You, Xiaohui Xie, and Zhangyang Wang, "Good students play big lottery better," *arXiv preprint arXiv:2101.03255*, 2021.

[61] Zhe Gan, Yen-Chun Chen, Linjie Li, Tianlong Chen, Yu Cheng, Shuohang Wang, and Jingjing Liu, "Playing lottery tickets with vision and language," *arXiv preprint arXiv:2104.11832*, 2021.

[62] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang, "A unified lottery ticket hypothesis for graph neural networks," *arXiv preprint arXiv:2102.06790*, 2021.

[63] Neha Mukund Kalibhat, Yogesh Balaji, and Soheil Feizi, "Winning lottery tickets in deep generative models," 2021.

[64] Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang, "Ultra-data-efficient gan training: Drawing a lottery ticket first, then training it toughly," *arXiv preprint arXiv:2103.00397*, 2021.

[65] Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang, "Long live the lottery: The existence of winning tickets in lifelong learning," in *International Conference on Learning Representations*, 2020.

[66] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[67] Wei Niu, Xiaolong Ma, Sheng Lin, Shihao Wang, Xuehai Qian, Xue Lin, Yanzhi Wang, and Bin Ren, "Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 907–922.

[68] Xiaolong Ma, Wei Niu, Tianyun Zhang, Sijia Liu, Sheng Lin, Hongjia Li, Wujie Wen, Xiang Chen, Jian Tang, Kaisheng Ma, et al., "An image enhancing pattern-based sparsity for real-time inference on mobile devices," in *European Conference on Computer Vision*. Springer, 2020, pp. 629–645.

[69] Jingyu Wang, Songming Yu, Zhuqing Yuan, Jinshan Yue, Zhe Yuan, Ruoyang Liu, Yanzhi Wang, Huazhong Yang, Xueqing Li, and Yongpan Liu, "Paca: A pattern pruning algorithm and channel-fused high pe utilization accelerator for cnns," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[70] Joost van Amersfoort, Milad Alizadeh, Sebastian Farquhar, Nicholas Lane, and Yarin Gal, "Single shot structured pruning before training," *arXiv preprint arXiv:2007.00389*, 2020.

[71] James E Falk and Jiming Liu, "On bilevel programming, part i: general nonlinear cases," *Mathematical Programming*, vol. 70, no. 1, pp. 47–72, 1995.

[72] Luis Vicente, Gilles Savard, and Joaquim Júdice, "Descent approaches for quadratic bilevel programming," *Journal of Optimization Theory and Applications*, vol. 81, no. 2, pp. 379–399, 1994.

[73] Douglas J White and G Anandalingam, "A penalty function approach for solving bi-level linear programs," *Journal of Global Optimization*, vol. 3, no. 4, pp. 397–419, 1993.

[74] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo, "On differentiating parameterized argmin and argmax problems with application to bi-level optimization," *arXiv preprint arXiv:1607.05447*, 2016.

[75] Shoham Sabach and Shimrit Shtern, "A first order method for solving convex bilevel optimization problems," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 640–660, 2017.

[76] Risheng Liu, Pan Mu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang, "A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6305–6315.

[77] Junyi Li, Bin Gu, and Heng Huang, "Improved bilevel model: Fast and optimal algorithm with theoretical guarantee," *arXiv preprint arXiv:2009.00690*, 2020.

[78] Saeed Ghadimi and Mengdi Wang, "Approximation methods for bilevel programming," *arXiv preprint arXiv:1802.02246*, 2018.

[79] Kaiyi Ji, Junjie Yang, and Yingbin Liang, "Bilevel optimization: Nonasymptotic analysis and faster algorithms," *arXiv preprint arXiv:2010.07962*, 2020.

[80] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang, "A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic," *arXiv preprint arXiv:2007.05170*, 2020.

[81] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil, "Forward and reverse gradient-based hyperparameter optimization," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1165–1173.

[82] Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo, "On the iteration complexity of hypergradient computation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3748–3758.

[83] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots, "Truncated back-propagation for bilevel optimization," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1723–1732.

[84] Yihua Zhang, Guanhuan Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang, and Sijia Liu, "Revisiting and advancing fast adversarial training through the lens of bi-level optimization," *arXiv preprint arXiv:2112.12376*, 2021.

[85] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine, "Meta-learning with implicit gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 113–124.

[86] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein, "Metapoison: Practical general-purpose clean-label data poisoning," *arXiv preprint arXiv:2004.00225*, 2020.

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

[88] Zhangyu Chen, Dong Liu, Xiaofei Wu, and Xiaochun Xu, "Research on distributed renewable energy transaction decision-making based on multi-agent bilevel cooperative reinforcement learning," 2019.

[89] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang, "Dsa: More efficient budgeted pruning via differentiable sparsity allocation," in *European Conference on Computer Vision*. Springer, 2020, pp. 592–607.

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

[91] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 184–199.

[92] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *Journal of Machine Learning Research*, vol. 22, no. 241, pp. 1–124, 2021.

[93] Hao Wang, Xiangyu Yang, Yuanming Shi, and Jun Lin, "A proximal iteratively reweighted approach for efficient network sparsification," *IEEE Transactions on Computers*, vol. 71, no. 1, pp. 185–196, 2020.

[94] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al., "Optimization with sparsity-inducing penalties," *Foundations and Trends® in Machine Learning*, vol. 4, no. 1, pp. 1–106, 2012.

[95] Uri Shaham, Yutaro Yamada, and Sahand Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," *arXiv preprint arXiv:1511.05432*, 2015.

[96] Xiang Deng and Zhongfei Mark Zhang, "Is the meta-learning idea able to improve the generalization of deep neural networks on the standard supervised learning?," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 150–157.

[97] Imre Csiszár, "Information geonetry and alternating minimization procedures," *Statistics and decisions*, vol. 1, pp. 205–237, 1984.

[98] Motasem Alfarra, Adel Bibi, Hasan Hammoud, Mohamed Gaafar, and Bernard Ghanem, "On the decision boundaries of neural networks: A tropical geometry perspective," *arXiv preprint arXiv:2002.08838*, 2020.

[99] Chelsea Finn, Pieter Abbeel, and Sergey Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[100] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[101] Ya Le and Xuan Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, pp. 3, 2015.

[102] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section .
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec C.4
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec C.4
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3 and Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sec. 4
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sec. 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Table A2
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] The licences of used datasets are provided in the cited references
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is included in the supplementary material
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Derivation of (3)

Based on the fact that the $\boldsymbol{\theta}^*(\mathbf{m})$ is satisfied with the stationary condition of the lower-level objective function in (2), we obtain

$$\nabla_{\boldsymbol{\theta}} g(\mathbf{m}, \boldsymbol{\theta}^*) = \nabla_{\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*) + \gamma \boldsymbol{\theta}^* = \mathbf{0}, \tag{A1}$$

where for ease of notation, we omit the dependence of $\boldsymbol{\theta}^*(\mathbf{m})$ w.r.t. $\mathbf{m}$. We then take derivative of the second equality of (A1) w.r.t. $\mathbf{m}$ by using the implicit function theory. This leads to

$$\begin{aligned} &\nabla^2_{\mathbf{m}\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*) + \frac{d\boldsymbol{\theta}^*(\mathbf{m})}{d\mathbf{m}} \nabla^2_{\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*) + \gamma \frac{d\boldsymbol{\theta}^*(\mathbf{m})}{d\mathbf{m}} = \mathbf{0}; \\ \Longrightarrow &\frac{d\boldsymbol{\theta}^*(\mathbf{m})}{d\mathbf{m}} = -\nabla^2_{\mathbf{m}\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*) [\nabla^2_{\boldsymbol{\theta}} \ell(\mathbf{m} \odot \boldsymbol{\theta}^*) + \gamma \mathbf{I}]^{-1}. \end{aligned} \tag{A2}$$

B BIP Algorithm Details

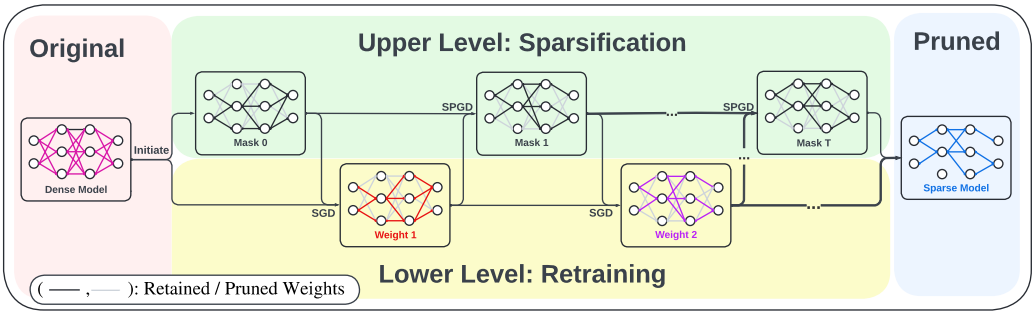


Figure A1: Overview of the BIP pruning algorithm. The BIP algorithm iteratively carry out model retraining in the lower level and pruning in the upper level. In the plots, SGD refers to the lower-level stochastic gradient descent update and SPGD refers to the upper-level stochastic projected gradient descent. Masks may vary between each iteration, and the pruned weights are indicated using the light gray color. Different colors of the edges in the neural networks refer to the weight update. The arcs in this figure represent the data flow of weights/connections.

At iteration t of BIP, there are two main steps:

★ *Lower-level SGD for model retraining*: Given $\mathbf{m}^{(t-1)}$, $\boldsymbol{\theta}^{(t-1)}$, and $\mathbf{z}^{(t-1)} := \mathbf{m}^{(t-1)} \odot \boldsymbol{\theta}^{(t-1)}$, we update $\boldsymbol{\theta}^{(t)}$ by applying SGD (stochastic gradient descent) to the lower-level problem of (1),

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \alpha \nabla_{\boldsymbol{\theta}} g(\mathbf{m}^{(t-1)}, \boldsymbol{\theta}^{(t-1)}) \overset{(7)}{=} \boldsymbol{\theta}^{(t-1)} - \alpha [\mathbf{m}^{(t-1)} \odot \nabla_{\mathbf{z}} \ell(\mathbf{z}) \,|_{\mathbf{z}=\mathbf{z}^{(t-1)}} + \gamma \boldsymbol{\theta}^{(t-1)}], \quad (\boldsymbol{\theta}\text{-step})$$

where $\alpha > 0$ is the lower-level learning rate.

★ *Upper-level SPGD for pruning*: Given $\mathbf{m}^{(t-1)}$, $\boldsymbol{\theta}^{(t)}$, and $\mathbf{z}^{(t+1/2)} := \mathbf{m}^{(t-1)} \odot \boldsymbol{\theta}^{(t)}$, we update $\mathbf{m}$ using PGD (projected gradient descent) along the IG-enhanced descent direction (2),

$$\begin{aligned} \mathbf{m}^{(t)} =& \mathcal{P}_{\mathcal{S}} \left[\mathbf{m}^{(t-1)} - \beta \frac{d\ell(\mathbf{m} \odot \boldsymbol{\theta}^{(t)})}{d\mathbf{m}} \,|_{\mathbf{m}=\mathbf{m}^{(t-1)}} \right] \\ \overset{(6)}{=} & \mathcal{P}_{\mathcal{S}} \left[\mathbf{m}^{(t-1)} - \beta \left(\boldsymbol{\theta}^{(t)} - \frac{1}{\gamma} \mathbf{m}^{(t-1)} \odot \nabla_{\mathbf{z}} \ell(\mathbf{z}) \,|_{\mathbf{z}=\mathbf{z}^{(t+1/2)}} \right) \odot \nabla_{\mathbf{z}} \ell(\mathbf{z}) \,|_{\mathbf{z}=\mathbf{z}^{(t+1/2)}} \right], \quad (\mathbf{m}\text{-step}) \end{aligned}$$

where $\beta > 0$ is the upper-level learning rate, and $\mathcal{P}_{\mathcal{S}}(\cdot)$ denotes the Euclidean projection onto the constraint set $\mathcal{S}$ given by $\mathcal{S} = \{\mathbf{m} \,|\, \mathbf{m} \in \{0,1\}^n, \mathbf{1}^T \mathbf{m} \leq k\}$ in (1) and is achieved by the top-k hard-thresholding operation as will be detailed below.

Implementation of discrete optimization. In the actual implementation, we use $\widetilde{\mathbf{m}}^{(t)} \in [0,1]^d$ and obtain $\mathbf{m}^{(t)} \in \{0,1\}^d$ where $\mathbf{m}^{(t)} \leftarrow \mathcal{P}_{\mathcal{S}} \left[\widetilde{\mathbf{m}}^{(t)} \right]$. The ($\mathbf{m}$-step) is then implemented as the following:

Table A1: Dataset and model setups. The following parameters are shared across all the methods.

Settings	CIFAR-10				CIFAR-100		Tiny-ImageNet	ImageNet
	RN-18	RN-20	RN-56	VGG-16	RN-18	RN-20	RN-18	RN-50
Batch Size	64	64	64	64	64	64	32	1024
Model Size	11.22 M	0.27 M	0.85 M	14.72 M	11.22 M	0.27 M	11.22 M	25.56 M

$$\widetilde{\mathbf{m}}^{(t)} = \widetilde{\mathbf{m}}^{(t-1)} - \beta \left(\boldsymbol{\theta}^{(t)} - \frac{1}{\gamma} \widetilde{\mathbf{m}}^{(t-1)} \odot \nabla_{\mathbf{z}} \ell(\mathbf{z}) \,|_{\mathbf{z}=\mathbf{z}^{(t+1/2)}} \right) \odot \nabla_{\mathbf{z}} \ell(\mathbf{z}) \,|_{\mathbf{z}=\mathbf{z}^{(t+1/2)}} \qquad (\widetilde{\mathbf{m}}\text{-step})$$

and then $\mathbf{m}^{(t)} \leftarrow \mathcal{P}_{\mathcal{S}} \left[\widetilde{\mathbf{m}}^{(t)} \right]$, with $\mathbf{z}^{(t+1/2)} := \mathbf{m}^{(t-1)} \odot \boldsymbol{\theta}^{(t)}$ as defined above.

C Additional Experimental Details and Results

C.1 Datasets and Models

Our dataset and model choices follow the pruning benchmark in [22]. We summarize the datasets and model configurations in Tab. A1. In particular, we would like to stress that we adopt the ResNet-18 with convolutional kernels of 3×3 in the first layer for Tiny-ImageNet, aligned with CIFAR-10 and CIFAR-100, compared to ImageNet (7×7). See `https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py` for more details.

C.2 Detailed Training Settings

Baselines. For both unstructured and structured pruning settings, we consider four baseline methods across various pruning categories, including IMP [17], OMP [17], HYDRA [9] and GRASP [23]. For HYDRA and GRASP, we adopt the original setting as well as hyper-parameter choices on their official code repositories. For IMP and OMP, we adopt the settings from the current SOTA implementations [22]. Details on the pruning schedules can be found in Tab. A2. In particular, HYDRA prunes the dense model to the desired sparsity with 100 epoch for pruning and 100 epoch for retraining. GRASP conducts one-shot pruning to the target sparsity, followed by the 200-epoch retraining. In each pruning iteration, IMP prunes 20% of the remaining parameters before 160-epoch retraining. HYDRA adopts the cosine learning rate scheduler for both pruning and retraining stage. The learning rate scheduler for IMP, OMP, and GRASP is the step learning rate with a learning rate decay rate of 0.1 at 50% and 75% epochs. The initial learning rate for all the methods are 0.1.

Hyper-parameters for BIP. In both structured and unstructured pruning settings, cosine learning rate schedulers are adopted, and BIP takes an initial learning rate of 0.1 for the upper-level problem (pruning) and 0.01 for the lower-level problem (retraining). The lower-level regularization coefficient λ is set to 1.0 throughout the experiments. By default, we only take one SGD step for lower-level optimization in all settings. Ablation studies on different SGD steps for lower-level optimization can be found in Fig. 6(b) and Fig. A13. We use 100 training epochs for BIP, and ablation studies on different training epochs for larger pruning ratios can be found in Fig. A12.

C.3 Additional Experiment Results

Comparison with IMP using reduced retraining epochs. As IMP is significantly more time-consuming than one-shot pruning methods, a natural way to improve the efficiency is to decrease the retraining epoch numbers at each pruning cycle. In Fig. A2, the performance and time consumption of IMP using 20, 40, and 80 epochs at each retraining cycle are presented. The results and conclusions are in general aligned with Fig. 2. First, with fewer epoch numbers, the time consumption decreases at the cost of evident performance degradation. Second, IMP with fewer epoch numbers are unable to obtain winning tickets. Thus, the direct simplification of IMP would hamper the pruning accuracy. This experiment shows the difficulty of achieving efficient and effective pruning under the scope of heuristics-based pruning, and thus justifies the necessity in developing a more powerful optimization-based pruning method.

Table A2: Computation complexities of different pruning methods on (CIFAR-10, ResNet-18) in unstructured pruning setting. The training epoch numbers of pruning/retraining baselines are consistent with their official settings or the latest benchmark implementations. All the evaluations are based on a single Tesla-V100 GPU.

Method \ Sparsity	Runtime v.s. targeted sparsity 20%	59%	83.2%	95.6%	Training epoch #
IMP	69 min	276 min	621 min	966 min	160 epoch retrain
GRASP	89 min				200 epoch retrain
OMP	69 min				160 epoch retrain
HYDRA	115 min				100 epoch prune 100 epoch retrain
BIP	86 min				100 epoch

Table A3: Detailed training details for each method. All the baselines adopt the recommended settings either from the official or their latest benchmark (*e.g.*, LTH[22]) for a fair comparison. Note by default setting, only our method BIP do not require additional epochs for retraining.

Method	Epoch Number	Initial Learning Rate	Learning Rate Scheduler	Learning Rate Decay Factor	Learning Rate Decay Epoch	Mementum	Weight Decay	Rewind Epoch	Warm-up
IMP	160 for Retrain	0.1	Step LR	10	80/120	0.9	5.00E-04	8	75 for VGG16
OMP	160 for Retrain	0.1	Step LR	10	80/120	0.9	5.00E-04	8	75 for VGG16
HYDRA	100 for Prune 100 for Retrain	0.1	Cosine LR	N/A	N/A	0.9	5.00E-04	N/A	75 for VGG16
GRASP	200 for Retrain	0.1	Step LR	10	100/150	0.9	5.00E-04	N/A	75 for VGG16
BIP	100	0.1 for $\mathbf{m}$; 0.01 for $\boldsymbol{\theta}$	Cosine LR	N/A	N/A	0.9	5.00E-04	N/A	75 for VGG16

Comparison with more prune-at-initialization baselines. In Fig. A3, we include more heuristics-based one-shot pruning baselines (SYNFLOW [24], SNIP [21]) for comparison. Together with GRASP, these methods belong to the category of pruning at initialization, which determines the sparse sub-networks prior to training. As we can see, the advantage of our method over the newly added methods are clear, and the benefit becomes more significant as the sparsity increases.

Hyper-parameter sanity check. In Fig. A4, we further demonstrate the influence of different choices of lower-level learning rate α as well as the batch size on the performance of BIP. Fig. A4 (a) shows that the test accuracy of BIP-pruned models is not quite sensitive to the choice of $\alpha \in \{0.01, 0.008\}$. A large α value (*e.g.*, $\alpha > 0.05$) will slightly decrease the performance of BIP. By contrast, a small α is preferred due to the fact that the model parameters are updated based on the pre-trained values. Fig. A4 (b) shows how the batch size influences the performance. As we can see, a large batch size might hurt the stochasticity of the algorithm and thus degrades the performance. We list our detailed batch size choices for different datasets in Tab. A1.

Additional structured pruning experiments. In addition to filter pruning in Fig. 4, we provide more results in the structured pruning setting, including both filter-wise and channel-wise pruning (as

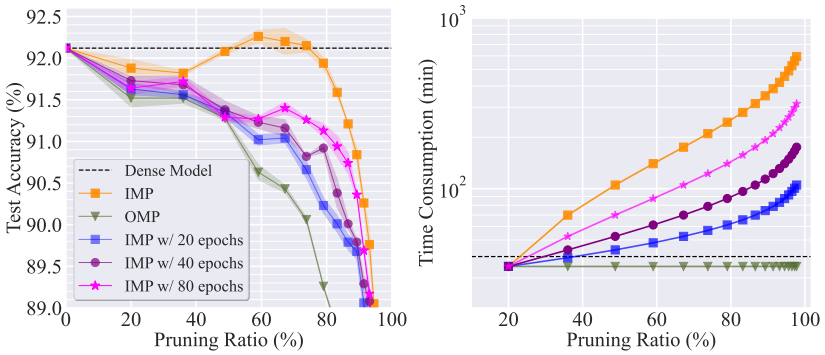


Figure A2: Performance comparisons among OMP, IMP, and IMP with less retraining epochs on CIFAR-10 with ResNet-20.

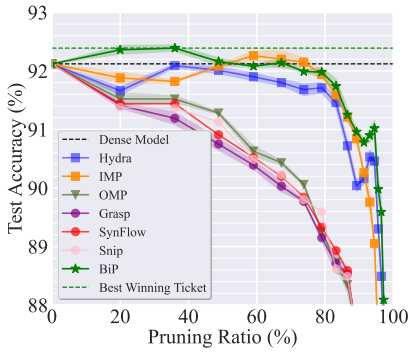
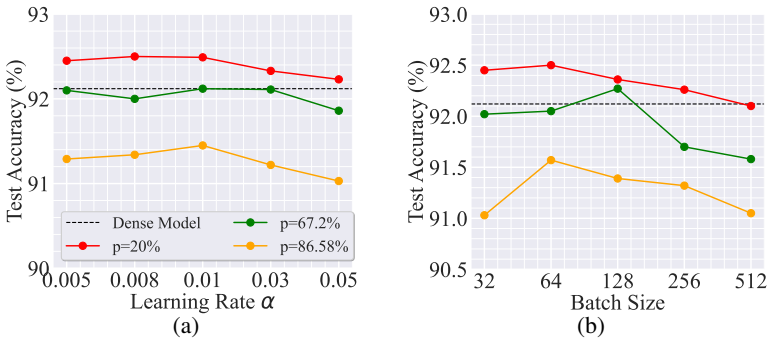
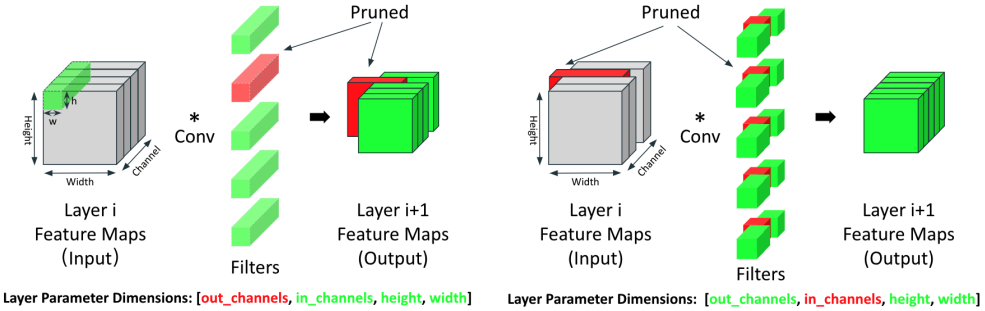


Figure A3: Unstructured pruning performance of BiP vs. prune-at-initialization baselines on (CIFAR-10, ResNet-20).



(a) (b)

Figure A4: Ablation studies of BiP on different hyper-parameters. All the experiments are based on CIFAR-10 with ResNet-20. We select three sparsity values of a wide range (from not sparse to extreme sparse) to make the results more general. We study the influence of different (a) lower-level learning rate α and (b) batch size.



(a) Filter-wise Pruning (b) Channel-wise Pruning

Figure A5: Illustration of filter-wise pruning and channel-wise pruning. The blocks in the middle column in (a) and (b) represent the parameters (filters) of the ith convolutional layer, where the red ones represent the pruning unit in each setting. The left blocks in gray denote the input feature maps and the right columns denote the output feature maps generated by the corresponding filters marked in the same color.

illustrated in Fig. A5). In Fig. A6, results on CIFAR-10 with VGG-16 and ResNet-56 are added as new experiments compared to Fig. 4. Fig. A7 shows the results of the channel-wise pruning. As we can see, consistent with Fig. 4, BiP is able to find the best winning tickets throughout the experiments while it is difficult for IMP to find winning tickets in most cases. We also notice that HYDRA, as the optimization-based baseline, serves as a strong baseline in filter-wise pruning. It also indicates the superiority of the optimization-based methods over the heuristics-based ones in dealing with more challenging pruning settings.

Training trajectory of BiP. We show in Fig. A8 that the BiP algorithm converges quite well within 100 training epochs using a cosine learning rate scheduler.

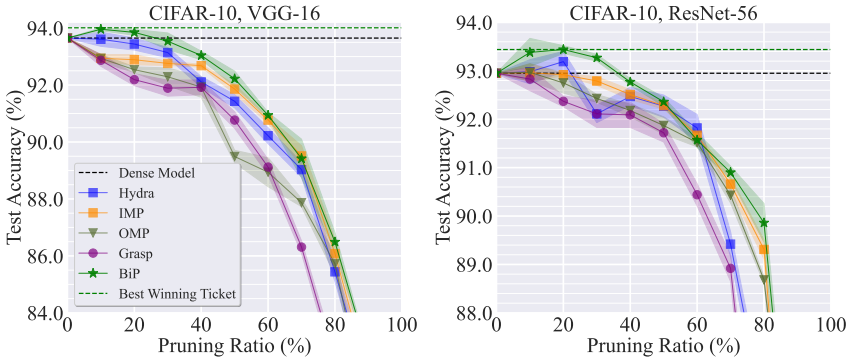


Figure A6: Filter-wise pruning test accuracy (%) v.s. sparsity (%) on CIFAR-10 with VGG-16 and ResNet-56.

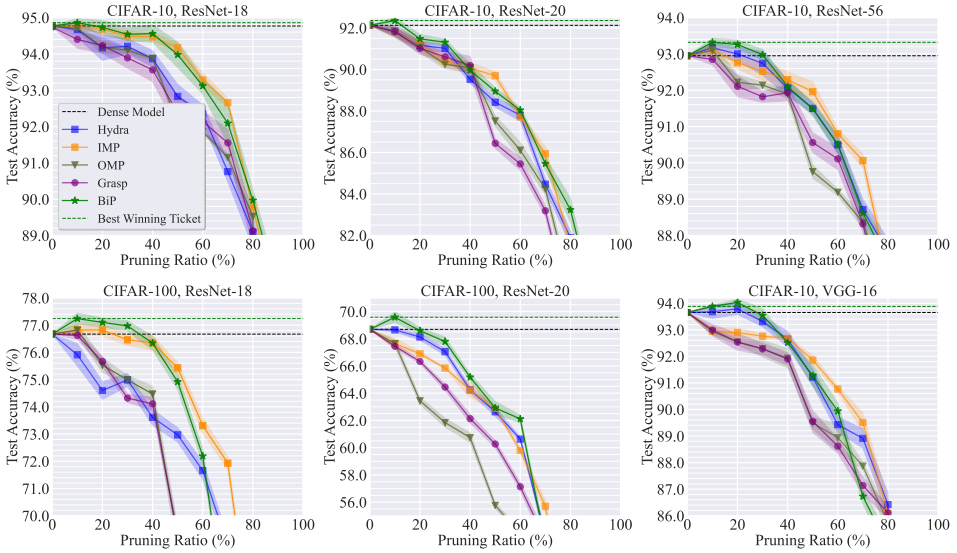


Figure A7: Channel-wise pruning test accuracy (%) v.s. sparsity (%). Settings are consistent with Fig. A6.

Experiments on unstructured pruning with more baselines. We compare our proposed method BiP to more baselines on different datasets and architectures in Fig. A9. We add two more baselines, including EarlyBird [28] and ProsPr [25]. The results show that ProsPr is indeed better than Grasp but is still not as good as IMP and our method BiP in different architecture-dataset combinations. Meanwhile, except for the unstructured pruning settings of ResNet18 pruning over CIFAR10 and CIFAR100, ProsPr, as a pruning before training, can achieve comparable performance to the state-of-the-art implementation of OMP. However, the gap between this SOTA pruning-at-initialization method and our method still exists. Besides, the result shows that EarlyBird can effectively achieve comparable or even better testing performance than OMP in most different architecture-dataset combinations, which is also the main contribution of [28]. However, EarlyBird is still not as strong as IMP in testing performance.

Experiments on structured pruning with more baselines. We compare our proposed method BiP to the new baseline ProsPr on different datasets and architectures in Fig. A10. on the structured pruning setting. As we can see, BiP consistently outperforms ProsPr and still stands top among all the baselines.

Ablation study on the sensitivity of BiP to rewinding epochs. We test the effects of rewinding + retraining of our proposed method BiP on more datasets and model architectures. Two conclusions can be drawn from Fig. A11. Firstly, the rewinding epoch number does not lead to significant improvements over the result without retraining (denoted as "N/A" in each sub-figure).

Figure A8: The training trajectory of BIP for unstructured pruning on (CIFAR-10, ResNet-18) with a pruning ratio of $p = 80\%$.

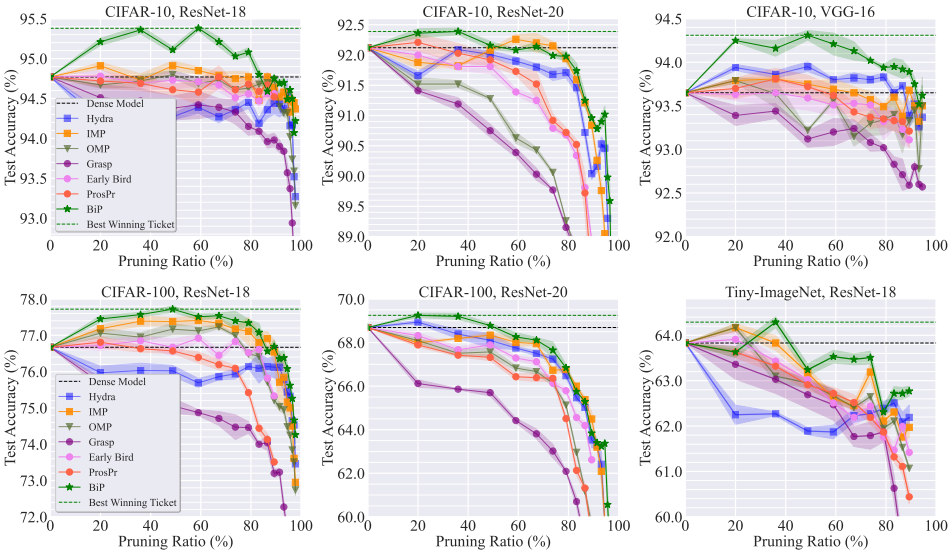


Figure A9: Unstructured pruning trajectory given by test accuracy (%) vs. pruning ratio (%). The visual presentation setting is consistent with Fig. 3. We consider two more baseline methods: EARLYBIRD [28] and PROSPR [25].

The effect of larger training epoch numbers with extreme sparsities. We allow more time (training epochs) for BIP when a higher pruning rate is considered and the results are shown in Fig. A12. Specifically, we test three datasets and consider three pruning ratios (p=86.58%, 94.50%, 97.75%). For each pruning ratio, we examine the test accuracy of BIP versus the training epoch number from 50 to 500. Note that the number of training epochs in our original experiment setup was set to 100. As we can see, the performance of BIP gets saturated when the epoch number is over 100.

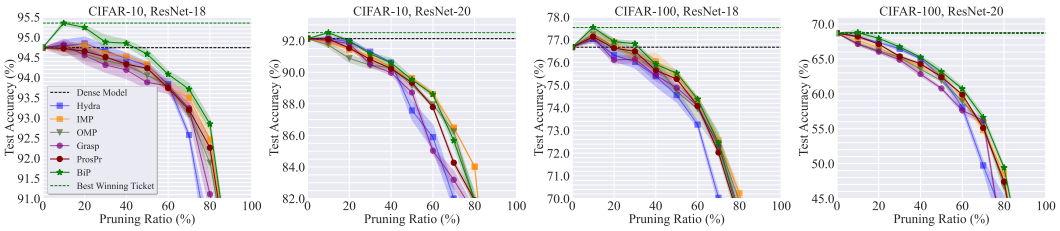


Figure A10: Structured pruning trajectory given by test accuracy (%) vs. pruning ratio (%). The visual presentation setting is consistent with Fig. 4. We add PROSPR [25] as our new baseline.

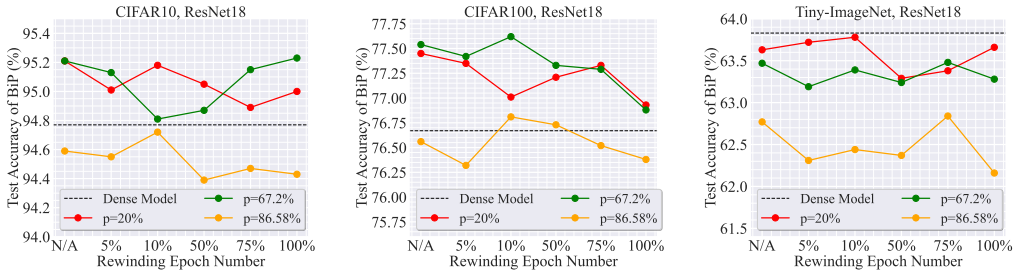


Figure A11: The sensitivity of BiP to rewinding epoch numbers on different datasets and model architectures. The experiment setting is consistent with Fig. 6(a).

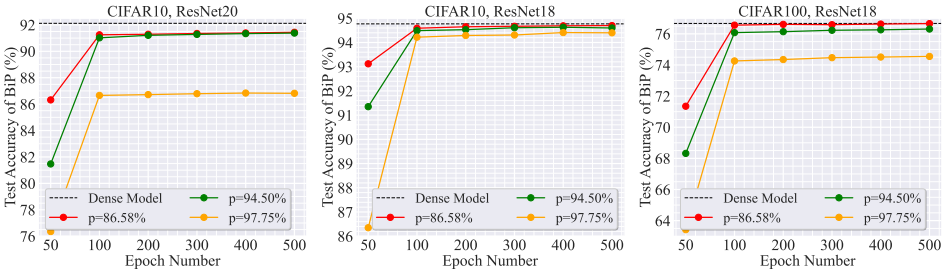


Figure A12: The effect of total training epoch number on the test accuracy with large pruning ratios. The epoch number is by default set to 100 in this paper. In each sub-figure, we report the performance of BiP with three different sparse ratios p.

Thus, even for a higher pruning ratio, the increase of training epoch number over 100 does not gain much improvement in accuracy.

The effect of different lower-level steps in BiP on the training dynamics. We conduct additional experiments to demonstrate the effectiveness of using one-step SGD in BiP. In our new experiments, we consider the number of SGD steps, 1, 3, and 5. We report the training trajectories of BiP in Fig. A13. As we can see, the use of multi-step SGD accelerates model pruning convergence at its early phase. Yet, if we run BiP for a sufficient number of epochs (we used 100 by default in other experiments), the final test accuracy of using different SGD settings shows little difference. Although the use of multiple SGD steps could improve the convergence speed, it introduces extra computation complexity per BLO step. Thus, from the overall computation complexity perspective, using 1 SGD step but running more epochs is advantageous in practice.

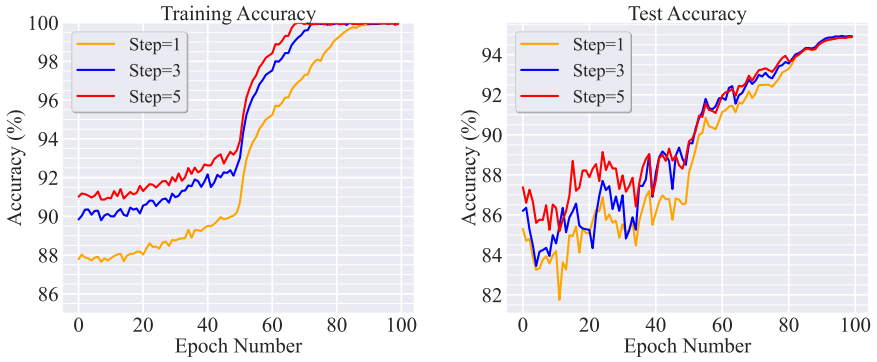


Figure A13: Training dynamics of BiP with different lower-level (SGD) steps on (CIFAR-10, ResNet-18) with the pruning ratio of p=80%.

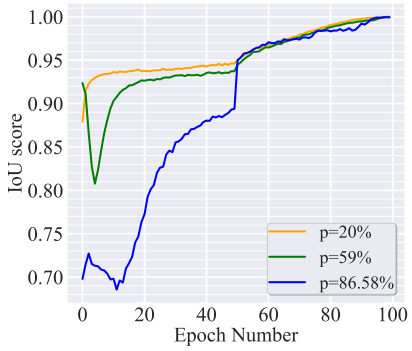


Figure A14: Training trajectory of the IoU (intersection over union) score between the masks of two adjacent epochs. We show the trajectory of different pruning ratios.

The training trajectory of the mask IoU score. To verify the argument that the mask also converges at the end of the training, we show the training trajectory of the mask similarity between two adjacent-epoch models in Fig. A14 at different pruning ratios. Here the mask similarity is represented through the intersection of the union (IoU) score of the two masks found by two adjacent epochs. The IoU score ranges from 0.0 to 1.0, and a higher IoU implies a larger similarity between the two masks. As we can see, the IoU score converges to 1.0 in the end, which denotes that the mask also converges at the end of the training phase. Also, with a smaller pruning ratio, the mask turns to converge more quickly.

The effect of different training batch schemes. We conducted ablation studies on three different schemes of BiP's training batches for the upper and lower level. In addition to two different random batches for the two levels, we also consider the same batch and the reverse batch scheme. BiP (same batch) always uses the same data batches for the two levels in each iteration while BiP (reverse batch) uses the data batches in a reversed order for the two level. Fig. A15 shows that both the random batch scheme (i.e., BiP) and the reverse batch scheme can bring a better testing accuracy performance than the same batch scheme throughout different pruning ratio settings. Fig. A16 shows that BiP the same batch scheme converges slower compared to the other two. Both of the results indicate BiP benefits from the diverse batch selection.

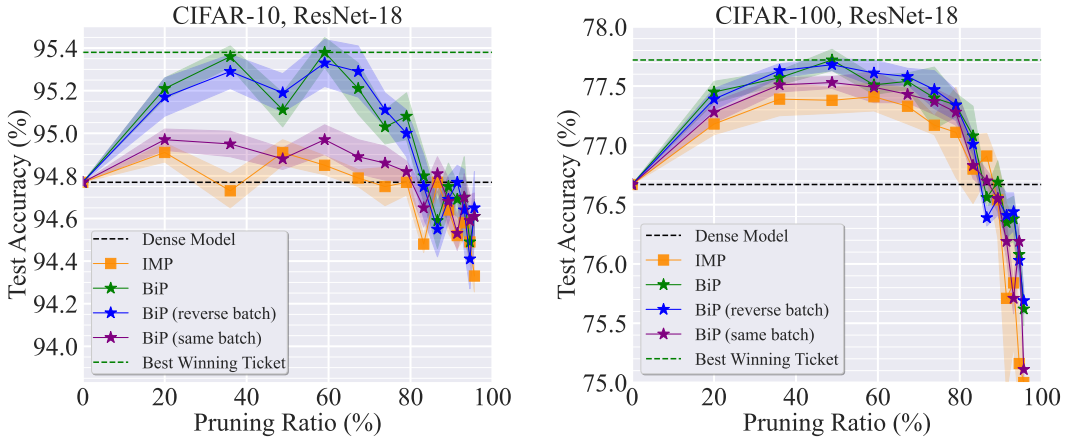


Figure A15: The effect of different training batch schemes on the performance of BiP. We consider two different variants of BiP denoted as BiP (reverse batch) and BiP (same batch). For BiP (reverse batch), the data batches are fed into the upper- and lower-step in a reversed order within each epoch, while for BiP (same batch), the data batches for upper- and lower-level are always the same. Experiment settings are consistent with Fig. 3. For better readability, we only plot the strongest baseline IMP for comparison.

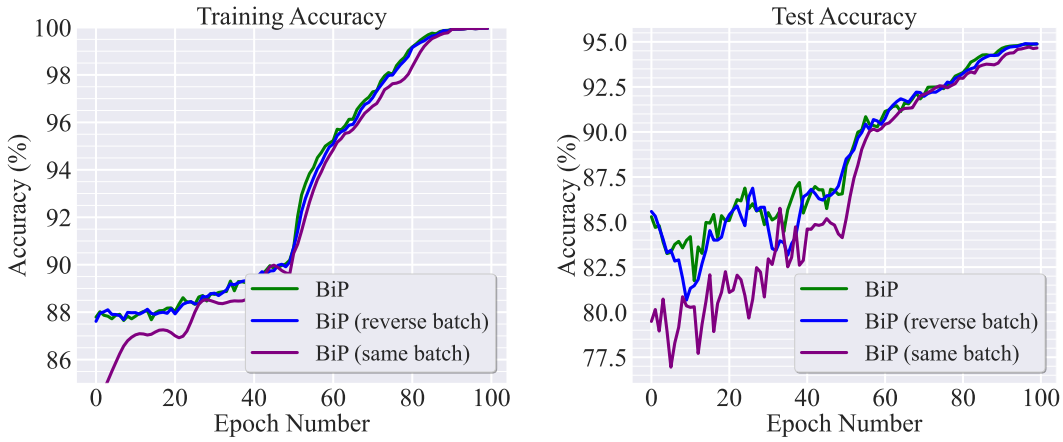


Figure A16: The effect of different training batch schemes on the training dynamics of BIP. We plot the training dynamics of different variants of BIP on (CIFAR-10, ResNet-18) with the pruning ratio of 80%.

C.4 Broader Impact

We do not recognize any potential negative social impacts of our work. Instead, we believe our work can inspire many techniques for model compression. The finding of structure-aware winning ticket also benefits the design of embedded solutions to deploying large-scale models on resource-limited edge devices (*e.g.*, FPGAs), providing broader impact on both scientific research and practical applications (*e.g.*, autonomous vehicles).