Auto-Train-Once: Controller Network Guided Automatic Network Pruning from Scratch

Xidong Wu*[1], Shangqian Gao*[1], Zeyu Zhang[2], Zhenzhen Li[3],
Runxue Bao[4], Yanfu Zhang[5], Xiaoqian Wang[6], Heng Huang[7]†
[1] University of Pittsburgh, [2] University of Arizona, [3] Bosch Center for AI, [4] GE HealthCare,
[5] College of William and Mary, [6] Purdue University, [7] University of Maryland College Park

Abstract

Current techniques for deep neural network (DNN) pruning often involve intricate multi-step processes that require domain-specific expertise, making their widespread adoption challenging. To address the limitation, the Only-Train-Once (OTO) and OTOv2 are proposed to eliminate the need for additional fine-tuning steps by directly training and compressing a general DNN from scratch. Nevertheless, the static design of optimizers (in OTO) can lead to convergence issues of local optima. In this paper, we proposed the Auto-Train-Once (ATO), an innovative network pruning algorithm designed to automatically reduce the computational and storage costs of DNNs. During the model training phase, our approach not only trains the target model but also leverages a controller network as an architecture generator to guide the learning of target model weights. Furthermore, we developed a novel stochastic gradient algorithm that enhances the coordination between model training and controller network training, thereby improving pruning performance. We provide a comprehensive convergence analysis as well as extensive experiments, and the results show that our approach achieves state-of-the-art performance across various model architectures (including ResNet18, ResNet34, ResNet50, ResNet56, and MobileNetv2) on standard benchmark datasets (CIFAR-10, CIFAR-100, and ImageNet). The code is available at https://github.com/xidongwu/AutoTrainOnce.

1. Introduction

Large-scale Deep Neural Networks (DNNs) have demonstrated remarkable prowess in various real-world applications [18, 32, 43, 44, 49, 69]. These large-scale networks leverage their substantial depth and intricate architecture to enhance approximations capability [51], capture intricate data features, and address a variety of computer vision tasks. Nevertheless, the large-scale models present a significant conflict with the hardware capability of devices during deployment since DNNs require substantial computational and storage overheads. To address this issue, model compression techniques [6, 17, 23, 57] have gained popularity to reduce the size of DNNs with minimal performance degradation and ease the deployment.

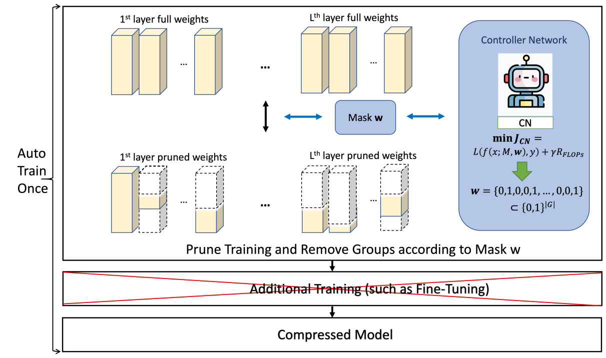

1st layer full weights
Lth layer full weights
Controller Network
CN
Mask w
$\min J_{CN} =$
$L(f(x; M, \boldsymbol{w}), y) + \gamma R_{FLOPs}$
$\boldsymbol{w} = \{0,1,0,0,1, \ldots, 0,0,1\} \subset \{0,1\}^{|G|}$
Auto
Train
Once
1st layer pruned weights
Lth layer pruned weights
Prune Training and Remove Groups according to Mask w
~~Additional Training (such as Fine-Tuning)~~
Compressed Model

Figure 1. Overview of Auto-Train-Once (ATO). The controller network generates mask $\mathbf{w}$ based on the size of ZIGs $\mathcal{G}$ to guide the automatic network pruning of the target model and we remove variable groups according to mask $\mathbf{w}$ after training. Additional training (such as fine-tuning) is not required after model training and we can directly get the final compressed model.

Structural pruning is a widely adopted direction to reduce the size of DNNs due to its generality and effectiveness [12]. Compared with weight pruning, structural pruning, particularly channel pruning, is more hardware-friendly, since it eliminates the need for post-processing steps to achieve computational and storage savings. Therefore, we focus on structural pruning for DNNs. However, it's worth noting that many existing pruning methods often come with notable limitations and require a complex multi-stage process. The process typically includes initial pre-

†This work was partially supported by NSF IIS 2347592, 2347604, 2348159, 2348169, DBI 2405416, CCF 2348306, CNS 2347617.

*These authors contributed equally to this work.

training, intermediate training for redundancy identification, and subsequent fine-tuning. Managing this multi-stage process of DNN training demands substantial engineering efforts and specialized expertise. To simplify the pruning methods, recent approaches like OTO [7] and OTOv2 [8] propose an end-to-end manner of training and pruning. These methods introduce the concept of zero-invariant groups (ZIGs), and simultaneously train and prune models without the reliance on further fine-tuning.

However, simple training frameworks in OTO and OTOv2 pose a challenge to model performance. They reformulate the objective as a constrained regularization problem. The local minima with better generalization may be scattered in diverse locations. Yet, as the augmented regularization in OTO penalizes the mixed L1/L2 norm of all trainable parameters in ZIGs, it restricts the search space to converge around the origin point. OTOv2 improves OTO by constructing pruning groups in ZIGs based on salience scores to penalize the trainable parameters only in pruning groups. However, model variables vary as training and the statically selected pruning groups of the optimizer in the early training stage can lead to convergence issues of local optima and poor final performance. Drawbacks in the algorithm design prevent them from giving a complete convergence analysis. For instance, OTO assumes the deep model as a strongly convex function and OTOv2 assumes a full gradient estimate at each iteration, which does not align with the practical settings of DNN training.

To enhance the model performance while maintaining a similar advantage, we propose Auto-Train-Once (ATO) and utilize a small portion of samples to train a network controller to dynamically manage the pruning operation on ZIGs. Experimental results demonstrate the success of our algorithm in identifying the optimal choice for ZIGs via the network controller.

In summary, the main contributions of this paper are summarized as follows:

1) We propose a generic framework to train and prune DNNs in a complete end-to-end and automatic manner. After model training, we can directly obtain the compressed model without additional fine-tuning steps.
2) We design a network controller to dynamically guide the channel pruning, preventing being trapped in local optima. Importantly, our method does not rely on the specific projectors compared with OTO and OTOv2. Additionally, we provide a comprehensive complexity analysis to ensure the convergence of our algorithm, covering both the general non-adaptive optimizer (e.g. SGD) and the adaptive optimizer (e.g. ADAM).
3) Empirical results show that our method overcomes the limitation arising from OTOs. Extensive experiments conducted on CIFAR-10, CIFAR-100, and ImageNet show that our method outperforms existing methods.

Table 1. summary of ATO and existing methods

Method	**ATO**	**OTOs**	**Others**
Training cost	**Low**	**Low**	**High**
Addition fine-tuning	**No**	**No**	**Yes**
Optimizer design	**Dynamic**	**Static**	**Static**
Convergence guarantee	$\surd$	$\not\surd$	-
Gradient projection	**General**	**(D)HSPG**	-

2. Related Works

To reduce the storage and computational costs, structural pruning methods identify and remove redundant structures from the full model. Most existing structural pruning methods adopt a three-stage procedure for pruning: (1) train a full model from scratch; (2) identify redundant structures given different criteria; (3) fine-tune or retrain the pruned model to regain performance. Different methods have different choices for the pruning criteria. Filter pruning [33] selects important structures with larger norm values. In addition to assessing channel or filter importance based on magnitude, the batch normalization scaling factor [26] can be used to identify important channels since batch normalization has gained popularity in the architecture of contemporary CNNs [18, 50]. Liu et al. [41] employ sparse regularization on the scaling factors of batch normalization to facilitate channel pruning. A channel is pruned if its associated scaling factor is deemed small. Structure sparse selection [25] extends the idea of using scaling factors of batch normalization to different structures, such as neurons, groups, or residual blocks, and sparsity regularization is also applied to these structures. Another line of research [13–15, 28, 29, 61] prunes unimportant channels through learnable scaling factors added for each structure. The learnable parameters are designed to be end-to-end differentiable, which enjoys the benefit of gradient-based optimizations. Inter-channel dependency [48] can also be used to remove channels. Greedy forward selection [60] iteratively adds channels with large norms to an empty model. In addition, reinforcement learning and evolutionary algorithms can also be used to select import structures. Automatic Model Compression (AMC) [20] uses a policy network to decide the width of each layer, and it is updated by policy gradient methods. In MetaPruning [35, 42], evolutionary algorithms are utilized to find the ideal combination of structures, and a hypernet generates the model weights. In addition to vision tasks, Natural Language Processing (NLP) has made significant advances across various tasks [52, 59, 64–68]. Concurrently, structure pruning has been increasingly applied to enhance the efficiency of large language models [54].

Regular structural pruning methods require manual efforts on all three stages, especially for the second and third stages. To unify all three stages and minimize manual efforts, OTO [7] formulates a structured-sparsity optimization problem and proposes the Half-Space Stochastic Projected Gradient (HSPG) to solve it. OTO resolves several disadvantages of previous methods based on structural sparsity [34, 55]: (1) multiple training stages since their group partition cannot isolate the impact of pruned structures on the model output; (2) heuristic post-processing to generate zero groups. On top of OTO, OTOv2 [8] introduces automated Zero-Invariant Groups (ZIGs) partition and Dual Half-Space Projected Gradient (DHSPG) to be user-friendly and better performant. Despite the advantages of OTOv2, it still has several problems: (1) the selection of pruning groups in ZIGs is static, once it is decided it can not be changed as model weights updates; (2) HSPG/DHSPG is more complex than simple proximal gradient operators. (3) The theoretical analysis of OTO and OTOv2 is not comprehensive and assumptions are overly strong. In this paper, we provide remedies for all the aforementioned weaknesses of OTOv2. A comprehensive comparison of ATO, OTO series, and other methods is shown in Table 1. It should be mentioned that our work is irrelevant to the gating modules used in dynamic pruning. The controller network builds a one-to-one mapping between an input vector and the resulting sub-network vector, and it is not designed to handle input features or images like dynamic pruning. The training process of ATO incorporates a regularization term to penalize useless structures decided by the controller network which is also not considered in dynamic pruning.

3. Proposed Method

The main idea of the method is to train a target network under the guidance of a trainable controller network. The controller network generates a mask $\mathbf{w}$, for each group in ZIGs [7]. As the training process concludes, the compression model is constructed by directly removing masked-out elements according to $\mathbf{w}$ without any more tuning.

3.1. Zero-Invariant Groups

Definition 1. *(Zero-Invariant Groups (ZIGs)) [7]. In the context of a layer-wise Deep Neural Network (DNN), entire trainable parameters are divided into disjoint groups $\mathcal{G} = \{g\}$. These groups are termed zero-invariant groups (ZIGs) when each group $g \in \mathcal{G}$ exhibits zero-invariant, where zero-invariant implies that setting all parameters in g to zero leads to the output corresponding to the next layer also being zeros.*

Chen et al. [7] firstly proposed the ZIGs. The Convolutional layer (Conv) without bias followed by the batch-normalization layer (BN) can be shown as below:

$$\mathcal{O}^l \leftarrow \mathcal{I}^l \otimes \hat{\mathcal{K}}^l, \mathcal{I}^{l+1} \leftarrow \frac{a\left(\mathcal{O}^l\right) - \boldsymbol{\mu}^l}{\boldsymbol{\sigma}^l} \odot \gamma^l + \boldsymbol{\beta}^l$$

where $\mathcal{I}^l$ denotes the input tensor, $\otimes$ denote the convolutional operation, $\mathcal{O}^l$ presents one output channel in l^{th} layer, $\odot$ is the element-wise multiplication, $a(\cdot)$ is the activation function, and $\boldsymbol{\mu}^l, \boldsymbol{\sigma}^l, \gamma^l, \boldsymbol{\beta}^l$ represent running mean, standard deviation, weight and bias, respectively in BN. Each output channel of the Conv $\hat{\mathcal{K}}^l$, and corresponding channel-wise BN weight γ^l and bias $\boldsymbol{\beta}^l$ belong to one ZIG because they being zeros results in their corresponding channel output to be zeros as well.

3.2. Controller Network

In the context of ZIGs, group-wise masks denoted as $\mathbf{w} \in \{0,1\}^N$ are generated by a Controller Network. The binary 0 and 1 represent the respective actions of removing and preserving channel groups. Controller Network incorporates bi-directional gated recurrent units (GRU) [9] followed by linear layers and Gumbel-Sigmoid [27] combined with straight-through estimator (STE) [5]. The utilization of the Gumbel-Sigmoid aims to produce a binary vector $\mathbf{w}$, that approximates a binomial distribution. The details of the Controller Network are in the supplementary materials.

With the mask $\mathbf{w}$, we can apply it to the feature maps to control the output of each group in ZIGs. For example, in a DNN, if the channels of l^{th} layer are in ZIGs and the weights of l^{th} layer can be written as $\mathcal{M}_l \in \Re^{C_l \times C_{l-1} \times k_l \times k_l}$, where C_l is the number of channels and k_l is the kernel size in l^{th} layer. The feature map of l^{th} layer can be represented by $\mathcal{F}_l \in \Re^{C_l \times W_l \times H_l}$, where H_l and W_l are height and width of the current feature map. With the mask $\mathbf{w}_l = \{0,1\}^{C_l}$ for l^{th} layer, the feature map of the l^{th} layer is then modified as $\widehat{\mathcal{F}_l} = \mathbf{w}_l \odot \mathcal{F}_l$. The definition of ZIGs shows setting the output as 0 for one ZIG is equal to setting all weights in this ZIG.

3.3. Auto-Train-Once

Here, we introduce our proposed algorithm, Auto-Train-Once (ATO). The details of ATO are shown in Algorithm 1.

First, we initiate the ZIG set (denoted as $\mathcal{G}$) by partitioning the trainable parameters of $\mathcal{M}$. Then, we build a controller network with model weight $\mathcal{W}$, configuring it in the way that the output dimension equals $|\mathcal{G}|$ ($|\cdot|$ is set cardinality). Subsequently, the controller network generates the model mask vector $\mathbf{w} = CN(\mathcal{W})$, and group in ZIGs $\mathcal{G}$ with mask value as 0 will be penalized in the project operation, as the Line 8 in Algorithm 1.

We can formulate the optimization problem with regularization as follows:

$$\begin{aligned} \min_{\mathcal{M}} \ \mathcal{J}(\mathcal{M}) :=& \mathcal{L}(\mathcal{M}) + g(\mathcal{M}) \\ =& \mathcal{L}\big(f(x;\mathcal{M}), y\big) + \sum_{g \in \mathcal{G}} \lambda_g \left\|[\mathcal{M}]_g\right\| \end{aligned} \quad (1)$$

where $f(x;\mathcal{M})$ is the output of target model with weight

$\mathcal{M}$, $\mathcal{L}\big(f(x;\mathcal{M}),y\big)$ is the loss function with data (x,y) and $\mathcal{G}$ is ZIGs. λ_g is a group-specific regularization coefficient and its value is decided by the output of the controller network, i.e. $\lambda_g = \lambda(1-[\mathbf{w}]_g)$. If λ_g is 0, then we do not put a penalty on the group g. After T_w warm-up steps, we add regularization to prune the target model. A larger λ typically results in a higher group sparsity [63]. To incorporate the group sparsity into optimization objective functions, there are several existing projection operators, such as Half-Space Projector (HSP) [7] as below:

$$\left[\mathrm{Proj}^{HS}_{\mathcal{S}_k}(\boldsymbol{z})\right]_g := \begin{cases} 0 & \text{if } [\boldsymbol{z}]_g^\top \left[\mathcal{M}\right]_g < \epsilon \left\|\left[\mathcal{M}\right]_g\right\|^2 \\ [\boldsymbol{z}]_g & \text{otherwise.} \end{cases} \tag{2}$$

and proximal gradient projector [63] as follows:.

$$\mathrm{prox}_{\eta\lambda_g}([\boldsymbol{z}]_g) = \begin{cases} [\boldsymbol{z}]_g - \eta\lambda_g \frac{[\boldsymbol{z}]_g}{\|[\boldsymbol{z}]_g\|_2}, \\ \text{ if } \|\ [\boldsymbol{z}]_g\ \| \geq \alpha\lambda_g, \\ 0,\ \text{otherwise .} \end{cases} \tag{3}$$

On the other hand, to avoid trapping in the local optima, we train a controller network to dynamically adjust the model mask from T_{start} to T_{end}. We use a small portion of training datasets $\mathcal{D}$ to construct $\mathcal{D}_{\text{CN}}$. The overall loss function for the controller network is as follows:

$$\begin{aligned} \min_{\mathcal{W}} \mathcal{J}_{\text{CN}}(\mathcal{W}) :=& \mathcal{L}\big(f(x;\mathcal{M},\mathbf{w}),y\big) \\ &+ \gamma\mathcal{R}_{\text{FLOPs}}(P(\mathbf{w}), pP_{\text{total}}) \end{aligned} \tag{4}$$

where $f(x;\mathcal{W},\mathbf{w})$ is the output of the target model with weight $\mathcal{W}$ based on model mask vector $\mathbf{w}$. $P(\mathbf{w})$ is the current FLOPs based on the mask $\mathbf{w}$, P_{total} is the total FLOPs of the original model, $p \in (0,1]$ is a hyperparameter to decide the target fraction of FLOPs, and γ is the hyper-parameter to control the strength of FLOPs regularization. The regularization term $\mathcal{R}_{\text{FLOPs}}$ is defined as:

$$\mathcal{R}_{\text{FLOPs}}(x,y) = \log(\max(x,y)/y)$$

In ATO, we train the target model and controller network alternately. After T_{end} epochs, we stop controller network training and freeze the model mask vector $\mathbf{w}$ to improve the stability of model training in the final phase.

4. Convergence and Complexity Analysis

In this section, we provide theoretical analysis to ensure the convergence of ATO to the solution of Eq. (1) in the manner of both theory and practice. Note: for convenience, we set z as the vector of network weight $\mathcal{M}$.

Algorithm 1 ATO Algorithm

```
Input: Target model with model weights $\mathcal{M}$ (no need to be pre-tained). Datasets $\mathcal{D}$, $\mathcal{D}_{\text{CN}}$, learning rate $\eta$, $\lambda$, $\gamma$, total steps $T$, warm-up steps $T_{\text{w}}$, controller network training steps $T_{start}$ and $T_{end}$
Initialization: Construct ZIGs $\mathcal{G}$ of $\mathcal{M}$. Build controller network with weight $\mathcal{W}$ based on the size of $\mathcal{G}$. $\mathbf{w}$ is initialized as $\{0,1\}^{|\mathcal{G}|}$
for $t = 1, 2 \ldots, T$ do
    for a mini-batch $(x, y)$ in $D$ do
        Compute the stochastic gradient estimator $\nabla_{\mathcal{M}}\mathcal{L}(\mathcal{M})$ in Eq. (1).
        Update model weights $\mathcal{M}$ with any stochastic optimizer.
        if $T \geq T_{\text{w}}$ then
            Perform projection operator and update following Eq. (2) or Eq. (3) on ZIGs with $\mathbf{w}$.
        end if
    end for
    if $T_{start} \leq T \leq T_{end}$ then
        $\mathcal{W}, \mathbf{w} \leftarrow$ CN-Update$(\mathcal{M}, \mathcal{W}, \mathbf{w}, \mathcal{D}_{\text{CN}})$
    end if
end for
Output: Directly remove pruned structures with mask $\mathbf{w}$ and construct a compressed model.
```

Algorithm 2 CN-Update$(\mathcal{M}, \mathcal{W}, \mathbf{w}, \mathcal{D}_{\text{CN}})$

```
Input: Target model with weights $\mathcal{M}$, controller network with weights $\mathcal{W}$, mask $\mathbf{w}$ and Datasets $\mathcal{D}_{\text{CN}}$, $\gamma$
for a mini-batch $(x, y)$ in $\mathcal{D}_{\text{CN}}$ do
    generate the mask $\mathbf{w}$ and calculate gradients estimator $\nabla_{\mathcal{W}}\mathcal{J}_{\text{CN}}(\mathcal{W})$ in Eq. (4).
    Update the controller network weight $\mathcal{W}$ with stochastic optimizer.
end for
Generate mask $\mathbf{w}$
Output: controller network $\mathcal{M}$ and $\mathbf{w}$
```

4.1. Stochastic Mirror Descent Method

We convert the algorithm into the stochastic mirror descent method to provide the convergence analysis, considering the stochastic non-adaptive optimizers (i.e., SGD) and stochastic adaptive optimizers (ADAM).

We set z as the vector of network weight $\mathcal{M}$, and define $\phi_t(z) = \frac{1}{2}z^T A_t z$, the Bregman divergence (i.e., Bregman distance) is defined as below:

$$D_\phi(z,x) = \phi(z) - \phi(x) - \langle \nabla\phi(x), z - x \rangle$$

To solve the general minimization optimization problem $\min_z f(z)$, the mirror descent method [4, 24] follows the below step:

$$z_{t+1} = \arg\min_{z} \left\{ f(z_t) + \langle \nabla f(z_t), z - z_t \rangle + \frac{1}{\eta} D_{\phi}(z, z_t) \right\}$$

where $\eta > 0$ is learning rate. It should be mentioned that the first two terms in the above function are a linear approximation of $f(z)$, and the last term is a Bregman distance between z and z_t. Most importantly, the constant terms $f(z_t)$ and $\langle \nabla f(z_t), z_t \rangle$ can be ignored in the above function. When choosing $\phi(z) = \frac{1}{2}\|z\|^2$, we have $D_{\phi}(z, z_t) = \frac{1}{2}\|z - z_t\|^2$. Then we have the standard gradient descent algorithm as below:

$$z_{t+1} = z_t - \eta \nabla f(z_t)$$

And the stochastic mirror descent update step as below:

$$z_{t+1} = \arg\min_{z} \left\{ \langle m_t, z \rangle + \frac{1}{\eta_t} D_{\phi_t}(z, z_t) + g_t(z) \right\}$$

where m_t is the momentum gradient estimator of $\nabla \mathcal{L}(z; \xi)$ and $m_t = (1 - \alpha_t) m_{t-1} + \alpha_t \nabla \mathcal{L}(z; \xi)$, η_t is the learning rate, $g(z)$ is a generally nonsmooth regularization in Eq. (1). The controller network uses the mask to adjust group-specific regularization coefficient λ_g in $g(z)$.

In the practice, since regularization $g(x)$ in composite functions Eq. (1) might not differentiable, we can minimize the loss function $\mathcal{L}$ firstly as step 6 in Algorithm 1, which is equivalent to the following generalized problem:

$$\tilde{z}_{t+1} = \arg\min_{z} \left\{ \langle m_t, z \rangle + \frac{1}{\gamma} D_t(z, z_t) \right\}$$

and then perform the projection operator as step 8 in Algorithm 1 as in Eq. (2) and Eq. (3) on ZIGs with $\mathbf{w}$.

For non-adaptive optimizer, we choose $\phi(z) = \frac{1}{2}\|z\|^2$, and $D_{\phi}(z, z_t) = \frac{1}{2}\|z - z_t\|^2$ and the mirror descent method will be reduced to the stochastic projected gradient descent method. For adaptive optimizer, we can generate the matrices A_t as in Adam-type algorithms [30], defined as

$$\tilde{v}_0 = 0,\ \tilde{v}_t = \beta \tilde{v}_{t-1} + (1 - \beta) \nabla_z \mathcal{L}(z_t; \xi_t)^2, \\ A_t = \mathrm{diag}(\sqrt{\tilde{v}_t} + \epsilon) \tag{5}$$

where $\tilde{v}$ is the second-moment estimator, and ϵ is a term to improve numerical stability in Adam-type optimizer. Then

$$D_t(z, z_t) = \frac{1}{2}(z - z_t)^T A_t (z - z_t). \tag{6}$$

4.2. Convergence Metrics and analysis

We introduce useful convergence metrics to measure the convergence of our algorithms. As in [16], we define a generalized projected gradient gradient mapping as:

$$\mathcal{P}_t = \frac{1}{\eta_t}(z_t - z^*_{t+1}), \tag{7}$$

$$z^*_{t+1} = \arg\min_{z} \left\{ \langle \nabla \mathcal{L}(z_t), z \rangle + \frac{1}{\eta_t} D_{\phi_t}(z, z_t) + g(z) \right\}$$

Therefore, for Problem (1), we use the standard gradient mapping metric $\mathbb{E}\|\mathcal{P}_t\|$ to measure the convergence of our algorithms

Finally, we present the convergence properties of our ATO algorithm under Assumptions 1 and 2. The following theorems show our main theoretical results. All related proofs are provided in the Supplement Material.

Theorem 1. *Assume that the sequence $\{z_t\}_{t=1}^T$ be generated from the Algorithm ATO (details of definition of variables are provided in the supplementary materials). When we have hyperparameters $\eta_t = \frac{\hat{c}}{(\bar{c}+t)^{1/2}}$, $\frac{\hat{c}}{\bar{c}^{1/2}} \leq \min\{1, \frac{\epsilon}{4L}\}$, $c_1 = \frac{4L}{\epsilon}, \alpha_{t+1} = c_1 \eta_t$, constant batch size $b = O(1)$, we have*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\|\mathcal{P}_t\| \leq \frac{\sqrt{G}\bar{c}^{1/4}}{T^{1/2}} + \frac{\sqrt{G}}{T^{1/4}} \tag{8}$$

where $G = \frac{4(\mathcal{J}(z_1) - \mathcal{J}(z^*))}{\epsilon \hat{c}} + \frac{2\sigma^2}{bL\epsilon\hat{c}} + \frac{2\bar{c}\sigma^2}{\hat{c}\epsilon L b} \ln(\bar{c} + T)$.

Remark 1. *(Complexity) To make the $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\,\|\mathcal{P}_t\| \leq \varepsilon$, we get $T = O(\varepsilon^{-4})$. Considering the use of constant batch size, $b = O(1)$, we have complexity $bT = O(\varepsilon^{-4})$, which matches the general complexity for stochastic optimizers [16] and guarantees the convergence of the proposed algorithm.*

5. Experiments

5.1. Setup

We assess the effectiveness of our algorithm through evaluations on image classification tasks, including datasets CIFAR-10 [31], CIFAR-100, and ImageNet [10] employing ResNet [18] and MobileNet-V2 [50] for comparison.

To compare with existing algorithms, we adjust the hyper-parameter p as in Eq. (4) to decide the final remaining FLOPs. A uniform setting with γ in Eq. (4) set to 4.0. The value of λ in Eq. (1) is set as 10 for different models and datasets. We set the start epoch of the controller network T_{start} at around 10% of the total training epochs, and the parameter T_{end} is set at 50% of the total training epochs. Detailed values are provided in supplementary materials. The general choice of T_{start} and T_{end} denotes that the training of the controller network is easy and robust. To mitigate the training costs arising from controller network training, we randomly sample 5% of the original dataset $\mathcal{D}$ to construct $\mathcal{D}_{\text{CN}}$, incurring only additional costs of less than 5% of the original training costs. We use ADAM [30] optimizer to train the controller network with an initial learning rate of 0.001. In addition, we use the proximal gradient project with l_2 norm in Eq. (3).

For the training of the target network, standard training recipes for ResNets on CIFAR-10, CIFAR-100, and ImageNet are followed, while for the MobileNet-V2, training

Table 2. Results comparison of existing algorithms on CIFAR-10 and CIFAR-100. Δ-Acc represents the performance changes relative to the baseline, and $+/-$ indicates an increase/decrease, respectively.

Dataset	Architecture	Method	Baseline Acc	Pruned Acc	Δ-Acc	Pruned FLOPs
CIFAR-10	ResNet-18	OTOv2 [8]	93.02 %	92.86%	-0.16%	79.7%
		ATO (ours)	94.41%	**94.51%**	+ **0.10%**	**79.8%**
	ResNet-56	DCP-Adapt [70]	93.80%	93.81%	+0.01%	47.0%
		SCP [28]	93.69%	93.23%	−0.46%	51.5%
		FPGM [21]	93.59%	92.93%	−0.66%	52.6%
		SFP [19]	93.59%	92.26%	−1.33%	52.6%
		FPC [22]	93.59%	93.24%	−0.25%	52.9%
		HRank [39]	93.26%	92.17%	−0.09%	50.0%
		DMC [13]	93.62%	92.69%	+0.07%	50.0%
		GNN-RL [62]	93.49%	93.59%	+0.10%	54.0%
		ATO (ours)	93.50%	**93.74%**	**+ 0.24%**	**55.0%**
		ATO(ours)	93.50%	93.48 %	−0.02%	**65.3%**
	MobileNetV2	Uniform [70]	94.47%	94.17%	−0.30%	26.0%
		DCP [70]	94.47%	94.69%	+0.22%	26.0%
		DMC [13]	94.23%	94.49%	+0.26%	40.0%
		SCOP [53]	94.48%	94.24%	-0.24%	40.3%
		ATO (ours)	94.45%	**94.78%**	**+0.33%**	**45.8%**
CIFAR-100	ResNet-18	OTOv2 [8]	-	74.96%	-	39.8%
		ATO (ours)	77.95%	**76.79%**	**−0.07%**	**40.1%**
	ResNet-34	OTOv2 [8]	-	76.31%	-	49.5%
		ATO (ours)	78.43 %	**78.54 %**	**+0.11%**	**49.5%**

settings in their original paper [50]. are utilize. The T_w is set as T_w around 20% of total epochs for all models and datasets. Due to the page constraints, detailed information on training can be found in the supplementary materials. The main counterpart of our algorithm is OTOv2, which also requires no additional fine-tuning. In addition, we also list other pruning algorithms.

5.2. CIFAR-10

For CIFAR-10, we select ResNet-18, ResNet-56 and MobileNetV2 as our target models. Table 2 presents the results of our algorithm (i.e., ATO) and other baselines on CIFAR-10.

ResNet-18. For ResNet-18, our algorithm achieves the best performance (in terms of Δ-Acc) compared to OTOv2 under the same pruned FLOPs. OTOv2 regresses 0.16% top-1 accuracy since it does not require a fine-tuning stage and uses the static pruning groups. Under the guidance of the controller network, we can select masks more precisely and overcome the limitations in OTOv2 which has better performance.

ResNet-56. In ResNet-56, our method shows better performance compared with baselines under similar pruned FLOPs. Specifically, our method does not rely on the fine-tuning stage and is more simple and more user-friendly. Furthermore, our algorithm outperforms the second-best algorithm GNN-RL by 0.14% according to Δ-Acc (ATO +0.24% vs. GNN-RL +0.10%) when pruning a little more FLOPs (ATO 55.0% vs. GNN-RL 54.0%). The gaps between other algorithms and ours are even larger.

MobileNet-V2. In MobileNet-V2, our method also has good performance. Our algorithm prunes most FLOPs (45.8%) and also achieves the best performance in terms of Δ-ACC (+0.33).

5.3. CIFAR-100

Our comparisons on CIFAR-100 involve ResNet-18 and ResNet-34. All results for the CIFAR-100 dataset are shown in Table 2.

Due to OTOv2 not reporting its result on CIFAR-100 with ResNet-18 and ResNet-34, we produce the results of OTOv2 on CIFAR-100 under the same setting as ours. Compared with results of OTOv2 in Table 2, our algorithm pruned a little more FLOAPs, while our algorithms improved the results largely.

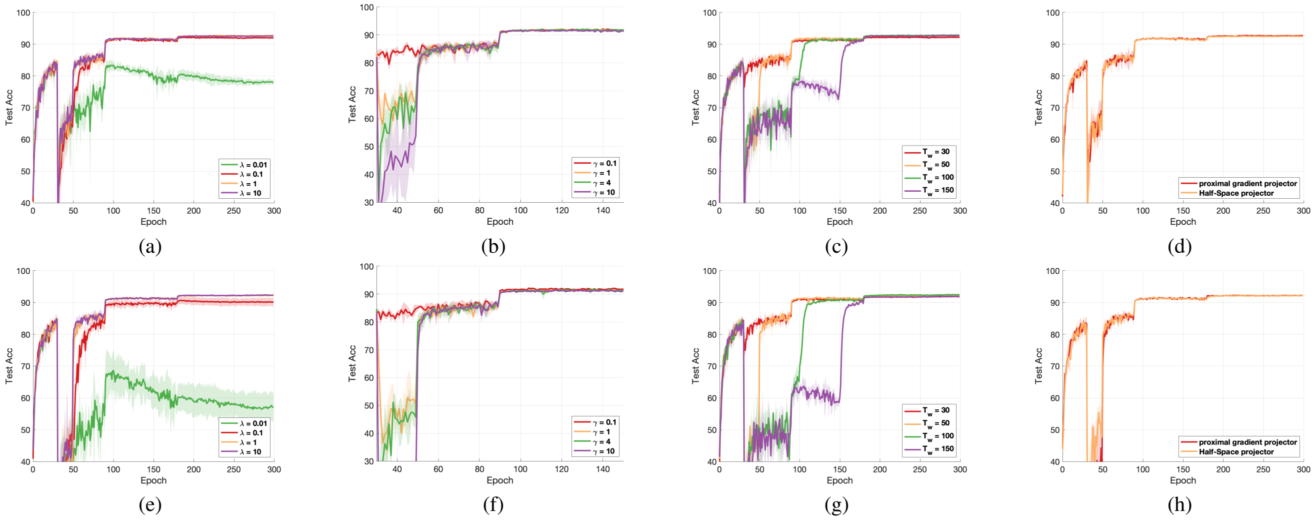
5.4. ImageNet

Subsequently, we employ ATO on ImageNet to demonstrate its effectiveness. In this context, we consider ResNet-34, ResNet-50, and MobileNetV2 as target models. The comparison between existing algorithms and ATO is presented in Table 3.

ResNet-34. In the ResNet-34, our algorithm achieves the best performance compared with others under the similar pruned FLOPs although our algorithm has a simpler training procedure. Our algorithm achieves 72.92% Top-1 accuracy and 91.15% Top-5 accuracy, which are better than other algorithms. SCOP and DMC prune similar FLOPs to our algorithm and have the same baseline results. Nonetheless, our method achieves a pruned Top-1 Accuracy that

Table 3. Comparison results on ImageNet with ResNet-34/50 and MobileNet-V2.

Architecture	Method	Base Top-1	Base Top-5	Pruned Top-1 (Δ Top-1)	Pruned Top-5 (Δ Top-5)	Pruned FLOPs
ResNet-34	FPGM [21]	73.92%	91.62%	72.63% (−1.29%)	91.08% (−0.54%)	41.1%
	Taylor [47]	73.31%	-	72.83% (−0.48%)	-	24.2%
	DMC [13]	73.30%	91.42%	72.57% (−0.73%)	91.11% (−0.31%)	43.4%
	SCOP [53]	73.31%	91.42%	72.62% (−0.69%)	90.98% (−0.44%)	**44.8%**
	ATO (ours)	73.31%	91.42%	**72.92% (−0.39%)**	**91.15% (−0.27%)**	44.1%
ResNet-50	DCP [70]	76.01%	92.93%	74.95% (−1.06%)	92.32% (−0.61%)	55.6%
	CCP [48]	76.15%	92.87%	75.21% (−0.94%)	92.42% (−0.45%)	54.1%
	FPGM [21]	76.15%	92.87%	74.83% (−1.32%)	92.32% (−0.55%)	53.5%
	ABCP [40]	76.01%	92.96%	73.86% (−2.15%)	91.69% (−1.27%)	54.3%
	DMC [13]	76.15%	92.87%	75.35% (−0.80%)	92.49% (−0.38%)	55.0%
	Random-Pruning [37]	76.15%	92.87%	75.13% (−1.02%)	92.52% (−0.35%)	51.0%
	DepGraph [11]	76.15%	-	75.83% (−0.32%)	-	51.7%
	DTP [38]	76.13%	-	75.55% (−0.58%)	-	**56.7%**
	ATO (ours)	76.13%	92.86%	**76.59% (+0.46%)**	**93.24% (+0.38%)**	55.2%
	DTP [38]	76.13%	-	75.24 % (−0.89%)	-	60.9%
	OTOv2 [8]	76.13%	92.86%	75.20% (−0.93%)	92.22% (−0.66%)	**62.6%**
	ATO (ours)	76.13%	92.86%	**76.07% (−0.06%)**	**92.92% (+0.06%)**	61.7%
	DTP [38]	76.13%	-	74.26% (−1.87%)	-	67.3%
	OTOv1 [7]	76.13%	92.86%	74.70% (−1.43%)	92.10% (−0.76%)	64.5%
	OTOv2 [8]	76.13%	92.86%	74.30% (−1.83%)	92.10% (−0.76%)	**71.5%**
	ATO (ours)	76.13%	92.86%	**74.77% (−1.36%)**	**92.25% (−0.61%)**	71.0%
MobileNet-V2	Uniform [50]	71.80%	91.00%	69.80% (−2.00%)	89.60% (−1.40%)	30.0%
	AMC [20]	71.80%	-	70.80% (−1.00%)	-	30.0%
	CC [36]	71.88%	-	70.91% (−0.97%)	-	28.3%
	MetaPruning [42]	72.00%	-	71.20% (−0.80%)	-	**30.7%**
	Random-Pruning [37]	71.88%	-	70.87% (−1.01%)	-	29.1%
	ATO (ours)	71.88%	90.29%	**72.02%** (+0.14%)	**90.19%** (−0.10%)	30.1%



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 2. (a, e): the impact of λ in regularization term in Eq. (1). (b, f): the effect of hyperparameter γ in $\mathcal{R}_{\text{FLOPs}}$ in Eq. (4). (c, g): the effect of T_{w}. (d, h): the effect of the project operation as in Eq. (2) and Eq. (2). Experiments are conducted on CIFAR-10 with ResNet-56 and $p = 0.45$ (a,b,c,d) and $p = 0.35$ (e,f,g,h).

is 0.30% and 0.35% superior to SCOP and DMC, respectively. Similar observations are made for Top-5 Accuracy, where our method outperforms SCOP and DMC by 0.17% and 0.04% respectively.

ResNet-50. In the ResNet-50, we report a performance portfolio under various pruned FLOPs, ranging from 55.2% to 71.0%. We compare with other counterparts in Figure 3. Although increasing pruned FLOPs and parameter reductions typically results in a compromise in accuracy, ATO exhibits a leading edge in terms of top-1 accuracy across various levels of FLOPs reduction. Compared with OTOv2, the results of our algorithm do not suffer from training simplicity. Especially, under pruned FLOPs of 61.7%, Top-1 acc of ATO achieves 76.07%, which is better than the OTOv2 by 0.87% under similar pruned FLOPs. In addition, even when pruned FLOPs is more than 70%, ATO still has good

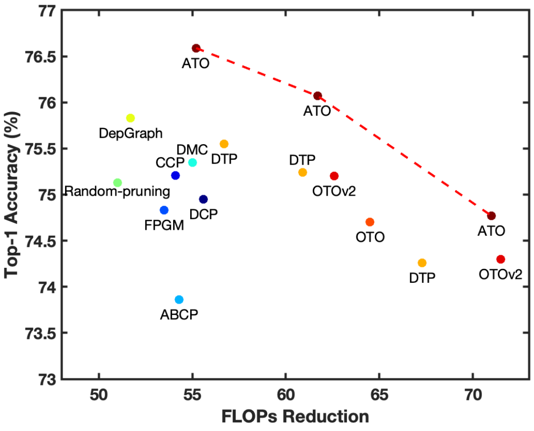


Figure 3. ResNet50 on ImageNet

performance compared with counterparts.

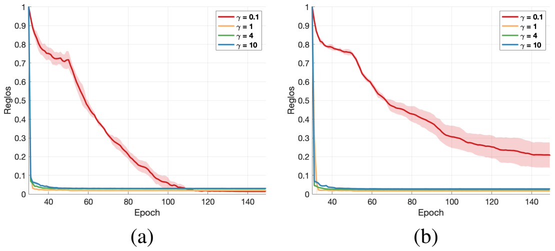
MobileNetv2. The lightweight model MobileNetv2 is generally harder to compress. Under the pruned FLOPs of 30%, our algorithm archives the best Top-1 acc compared with other methods, even though our algorithm has simpler training procedures. Our algorithm achieves 72.02% Top-1 accuracy and 90.19% Top-5 accuracy while the results of other counterparts are below the baseline results.

5.5. Ablation Study

We study the impact of different hyperparameters on model performance. We use the RenNet-56 on CIFAR-10. Note that the falling gap at Epoch 30 is because controller network training starts at Epoch 30 and then we test model performance under the mask vector $\mathbf{w}$, which is equivalent to removing the corresponding ZIGs.

The impact of λ. We study the impact of regularization coefficient λ in Eq. (1), and we plot the test accuracy in Figure 2a and Figure 2e. From the curves, we can see that λ plays an important role in the model training. If the value of λ is too small, it will harm model performance, especially when pruned FLOPs are large ($p = 0.35$).

The impact of γ. We plot the impact of hyper-parameter controlling γ, which decide the strength of FLOPs regularization in Eq. (4) and we plot the test accuracy in Figure 2b and Figure 2f. From the results, we can see that accuracy is lightly affected by γ and the curves of different γ converge quickly after the model starts training (Epoch = 30). Furthermore, we plot $\mathcal{R}_{\text{FLOPs}}$ in Figure 4, and the controller network under different γ converges before training stops at Epoch =150 with different speeds. Note the loss value is scaled to [0, 1]. It shows that selecting a too-small γ may hinder the controller network from achieving the target FLOPs when the pruning rate is large. Otherwise, the training of the controller network is robust.



(a)

(b)

Figure 4. the effect of hyperparameter γ in $\mathcal{R}_{\text{FLOPs}}$ in Eq. (4). Experiments are conducted on CIFAR-10 with ResNet-56 with $p = 0.45$ (a) and $p = 0.35$ (b).

The impact of T_w. We study the impact of T_w when training target model in Figure 2c and Figure 2g. Since we start training the controller network at Epoch 30 and we need a mask $\mathbf{w}$ from the controller network, T_w is selected from set $\{30, 50, 100, 150\}$. In general, they can converge to the optimal under the mask with different T_w.

The impact of projection operator. To verify our proposed is algorithm flexible to the projection operator, we plot the test accuracy of the proximal gradient projector in 3 and Half-Space projector 2. In OTO [7] and OTOv2 [8], due to the limitation of the manually selected static mask, they have to use (D)HSPG [8] to achieve better performance. In the test, we show that the controller network helps us solve these tough questions and there is no big difference between the two projection operators. Due to the easy implementation and the better efficiency of the proximal gradient projector, we use it in our experiments.

6. Conclusion

In this paper, we investigate automatic network pruning from scratch and address the limitations found in existing algorithms, including 1) complex multi-step training procedures and 2) the suboptimal outcomes associated with statically chosen pruning groups. Our solution, Auto-Train-Once (ATO), introduces an innovative network pruning algorithm designed to automatically reduce the computational and storage costs of DNNs without reliance on the extra fine-tuning step. During the model training phase, a controller network dynamically generates the binary mask to guide the pruning of the target model. Additionally, we have developed a novel stochastic gradient algorithm that offers flexibility in the choice of projection operators and enhances the coordination between the model training and the controller network training, thereby improving pruning performance. In addition, we present a theoretical analysis under mild assumptions to guarantee convergence, along with extensive experiments. The experiment results demonstrate that our algorithm achieves state-of-the-art performance across various model architectures, including ResNet18, ResNet34, ResNet50, ResNet56, and MobileNetv2 on standard benchmark datasets such as CIFAR-10, CIFAR-100, as well as ImageNet.

References

[1] Runxue Bao, Bin Gu, and Heng Huang. Fast oscar and owl regression via safe screening rules. In *International conference on machine learning*, pages 653–663. PMLR, 2020. 13

[2] Runxue Bao, Bin Gu, and Heng Huang. An accelerated doubly stochastic gradient method with faster explicit model identification. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 57–66, 2022. 13

[3] Runxue Bao, Xidong Wu, Wenhan Xian, and Heng Huang. Doubly sparse asynchronous learning for stochastic composite optimization. In *Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1916–1922, 2022. 13

[4] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003. 4

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

[6] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006. 1

[7] Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. *Advances in Neural Information Processing Systems*, 34:19637–19651, 2021. 2, 3, 4, 7, 8

[8] Tianyi Chen, Luming Liang, Tianyu Ding, Zhihui Zhu, and Ilya Zharkov. Otov2: Automatic, generic, user-friendly. *arXiv preprint arXiv:2303.06862*, 2023. 2, 3, 6, 7, 8, 12

[9] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 3, 12

[10] 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*, pages 248–255. Ieee, 2009. 5

[11] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101, 2023. 7

[12] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019. 1

[13] Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. Discrete model compression with resource constraint for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1899–1908, 2020. 2, 6, 7

[14] Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9270–9280, 2021.

[15] Shangqian Gao, Zeyu Zhang, Yanfu Zhang, Feihu Huang, and Heng Huang. Structural alignment for network pruning through partial regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17402–17412, 2023. 2

[16] Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305, 2016. 5, 13

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

[18] 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*, pages 770–778, 2016. 1, 2, 5

[19] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2234–2240, 2018. 6

[20] 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)*, pages 784–800, 2018. 2, 7

[21] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 6, 7

[22] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2009–2018, 2020. 6

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

[24] Feihu Huang, Xidong Wu, and Heng Huang. Efficient mirror descent ascent methods for nonsmooth minimax problems. *Advances in Neural Information Processing Systems*, 34:10431–10443, 2021. 4, 13

[25] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018. 2

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, pages 448–456. JMLR.org, 2015. 2

[27] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 3

[28] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *International Conference on Machine Learning*, pages 5122–5131. PMLR, 2020. 2, 6

[29] Jaedeok Kim, Chiyoun Park, Hyun-Joo Jung, and Yoonsuck Choe. Plug-in, trainable gate for streamlining arbitrary neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 2

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5, 13

[31] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5

[32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 1

[33] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ICLR*, 2017. 2

[34] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8018–8027, 2020. 3

[35] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 608–624. Springer, 2020. 2

[36] Yuchao Li, Shaohui Lin, Jianzhuang Liu, Qixiang Ye, Mengdi Wang, Fei Chao, Fan Yang, Jincheng Ma, Qi Tian, and Rongrong Ji. Towards compact cnns via collaborative compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6438–6447, 2021. 7

[37] Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. Revisiting random channel pruning for neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 191–201, 2022. 7

[38] Yunqiang Li, Jan C van Gemert, Torsten Hoefler, Bert Moons, Evangelos Eleftheriou, and Bram-Ernst Verhoef. Differentiable transportation pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16957–16967, 2023. 7

[39] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 6

[40] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 673 – 679, 2020. 7

[41] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 2

[42] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3296–3305, 2019. 2, 7

[43] Xiaobo Ma, Abolfazl Karimpour, and Yao-Jan Wu. Statistical evaluation of data requirement for ramp metering performance assessment. *Transportation Research Part A: Policy and Practice*, 141:248–261, 2020. 1

[44] Xiaobo Ma, Abolfazl Karimpour, and Yao-Jan Wu. Eliminating the impacts of traffic volume variation on before and after studies: a causal inference approach. *Journal of Intelligent Transportation Systems*, pages 1–15, 2023. 1

[45] Yongsheng Mei, Mahdi Imani, and Tian Lan. Bayesian optimization through gaussian cox process models for spatio-temporal data. *arXiv preprint arXiv:2401.14544*, 2024. 13

[46] Yongsheng Mei, Hanhan Zhou, and Tian Lan. Projection-optimal monotonic value function factorization in multi-agent reinforcement learning. In *Proceedings of the 2024 International Conference on Autonomous Agents and Multiagent Systems*, 2024. 13

[47] 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*, pages 11264–11272, 2019. 7

[48] Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In *International Conference on Machine Learning*, pages 5113–5122, 2019. 2, 7

[49] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[50] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 2, 5, 6, 7, 12

[51] Zuowei Shen. Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, 28(5):1768–1811, 2020. 1

[52] Hannah Smith, Zeyu Zhang, John Culnan, and Peter Jansen. ScienceExamCER: A high-density fine-grained science-domain corpus for common entity recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4529–4546, Marseille, France, 2020. European Language Resources Association. 2

[53] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *Advances in Neural Information Processing Systems*, 33, 2020. 6, 7

[54] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural*

Language Processing (EMNLP), pages 6151–6162, Online, 2020. Association for Computational Linguistics. 2

[55] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016. 3

[56] Xidong Wu, Feihu Huang, Zhengmian Hu, and Heng Huang. Faster adaptive federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10379–10387, 2023. 13

[57] Xidong Wu, Wan-Yi Lin, Devin Willmott, Filipe Condessa, Yufei Huang, Zhenzhen Li, and Madan Ravi Ganesh. Leveraging foundation models to improve lightweight clients in federated learning. *arXiv preprint arXiv:2311.08479*, 2023. 1

[58] Xidong Wu, Jianhui Sun, Zhengmian Hu, Aidong Zhang, and Heng Huang. Solving a class of non-convex minimax optimization in federated learning. *Advances in Neural Information Processing Systems*, 36, 2024. 13

[59] Dongfang Xu, Zeyu Zhang, and Steven Bethard. A generate-and-rank framework with semantic type regularization for biomedical concept normalization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8452–8464, Online, 2020. Association for Computational Linguistics. 2

[60] Mao Ye, Chengyue Gong, Lizhen Nie, Denny Zhou, Adam Klivans, and Qiang Liu. Good subnetworks provably exist: Pruning via greedy forward selection. In *International Conference on Machine Learning*, pages 10820–10830. PMLR, 2020. 2

[61] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 2130–2141, 2019. 2

[62] Sixing Yu, Arya Mazaheri, and Ali Jannesari. Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In *International Conference on Machine Learning*, pages 25656–25667. PMLR, 2022. 6

[63] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67, 2006. 4

[64] Zeyu Zhang and Steven Bethard. Improving toponym resolution with better candidate generation, transformer-based reranking, and two-stage resolution. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 48–60, Toronto, Canada, 2023. Association for Computational Linguistics. 2

[65] Zeyu Zhang, Thuy Vu, and Alessandro Moschitti. Joint models for answer verification in question answering systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3252–3262, Online, 2021. Association for Computational Linguistics.

[66] Zeyu Zhang, Thuy Vu, Sunil Gandhi, Ankit Chadha, and Alessandro Moschitti. Wdrass: A web-scale dataset for document retrieval and answer sentence selection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, page 4707–4711, New York, NY, USA, 2022. Association for Computing Machinery.

[67] Zeyu Zhang, Thuy Vu, and Alessandro Moschitti. In situ answer sentence selection at web-scale. *arXiv preprint arXiv:2201.05984*, 2022.

[68] Zeyu Zhang, Thuy Vu, and Alessandro Moschitti. Double retrieval and ranking for accurate question answering. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1751–1762, Dubrovnik, Croatia, 2023. Association for Computational Linguistics. 2

[69] Hanhan Zhou, Tian Lan, Guru Prasadh Venkataramani, and Wenbo Ding. Every parameter matters: Ensuring the convergence of federated learning with dynamic heterogeneous models reduction. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[70] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018. 6, 7

7. Implementation Details

In this section, we provide more details of the implementation.

We follow the standard training example in the training on CIFAR-10 and CIFAR-100 datasets. The model is trained for 300 epochs. We select SGD as the optimizer with a learning rate of 0.1, a momentum of 0.9, and a weight decay of 10^{-4}. The value of p in Eq. (4) for all datasets and models are presented in Table 5.

Table 4. Choice of p in Eq. (4) for different datasets

DataSet	Model	p
CIFAR-10	ResNet-18	0.1908
	ResNet-56	0.3500 / 0.4500
	MobileNetV2	0.4900
CIFAR-100	ResNet-18	0.5952
	ResNet-34	0.5020
ImageNet	ResNet-34	0.5400
	ResNet-50	0.3700/0.2960/0.1930
	MobileNetV2	0.6578

To train ResNet models on ImageNet, we follow the ImageNet training setting [*] in OTOv2 [8] and train the ResNet models for 240 epochs. For MobileNet-V2, we train the model for 300 epochs with the cos-annealing learning rate scheduler and a start learning rate of 0.05, and weight decay 4×10^{-5} as mentioned in their original paper [50]. As mentioned in the paper, we set T_{start} to around 10% of the total training epochs and T_{end} to 50% of the total training epochs.

Table 5. Architecture of Controller Network

Layer Type	Shape
Input	$\lvert\mathcal{B}\rvert \times 64$
Bi-GRU	$64 \times 128 \times 2$
LayerNorm + ReLU	256
Linear Layer	$256 \times \lvert B_i\rvert, B_i \subset \mathcal{B}$
Concatenate	$\lvert\mathcal{G}\rvert$
Gumbel-Sigmoid	-
Round $\rightarrow$ $\mathbf{w}$	-

Table 5 shows the architecture of the controller network. Controller Network uses bi-directional gated recurrent units (GRU) [9] followed by linear layers with output as o, where the dimension of o is equal to the size of ZIGs $\mathcal{G}$. To reduce the training costs, we divide ZIGs $\mathcal{G}$ into multiple disjoint blocks $\mathcal{B} = \{B_1, \cdots, B_{|\mathcal{B}|}\}$ ($B_1 \cup B_2 \cup \cdots B_{|\mathcal{B}|} = \mathcal{G}$ and $\sum |B_i| = |\mathcal{G}|$), and each block is one layer in ResNet models or one InvertedResidual block in MobileNetv2. Finally, the linear layer projects the output to the $|B_i|$ and $\sum |B_i| = |\mathcal{G}|$.

The model mask vector is generated as below:

$$\mathbf{w} = \text{round}(\text{sigmoid}((o + s + b)/\tau))$$

where sigmoid$(\cdot)$ is the sigmoid function, round$(\cdot)$ is the rounding function, s is sampled from Gumbel distribution ($s \sim$ Gumbel$(0, 1)$), b and τ are constants with values as 3.0 and 0.4.

8. Theoretical analysis

In this section, we provide a detailed convergence analysis of our algorithms. We define the z as the vector of the target model weight and we reformulated our algorithm (i.e., ATO) in the mirror descent format, as presented in Algorithm 3, to facilitate analysis. It should be noted that t in Algorithm 3 denotes the update iteration steps to discuss the training progression in each step, instead of the training epoch index as before. $g(z) = \sum_{g \in \mathcal{G}} \lambda_g \left\| [\mathcal{M}]_g \right\|$ and $g(z)$ varies as training. In the theoretical analysis, we assume the change of $g(z)$ is slow and regard it as static.

[*] https://github.com/tianyic/only_train_once/blob/main/tutorials/02.resnet50_imagenet.ipynb

As discussed in the paper, we define a Bregman distance [3, 16, 24] associated with $\phi(z)$ as follows:

$$D_\phi(z', z) = \phi(z') - \big[\phi(z) + \langle \nabla\phi(z), z' - z\rangle\big], \quad \forall z, z' \tag{9}$$

Given that $\phi(z) = \frac{1}{2}z^T Az$ in our paper, we have $D_\phi(z', z) = \frac{1}{2}(z' - z)^T A_t(z' - z)$. For the non-adaptive optimizer (e.g., SGD), we choose $A = I$. For adaptive optimizer (i.e., Adam), we update A_t as in Adam-type algorithms [30], defined as

$$\begin{aligned}&\tilde{v}_0 = 0,\ \tilde{v}_t = \beta\tilde{v}_{t-1} + (1-\beta)\nabla_z\mathcal{L}(z_t;\xi_t)^2,\\ &A_t = \text{diag}(\sqrt{\tilde{v}_t} + \epsilon)\end{aligned}$$

where $\tilde{v}$ is the second-moment estimator, and ϵ is a term to improve numerical stability in Adam-type optimizer. Therefore, $\phi(z)$ is a ϵ-strongly convex function. We first give some useful assumptions and lemmas.

Assumption 1. *(Unbiased gradient and bounded variance) Set $\xi = (x, y)$ as a batch of samples, z as the vector of network weight $\mathcal{M}$ and $\mathcal{J}(z) = \mathcal{L}(z) + g(z)$. Assume loss function $\mathcal{L}(z;\xi)$ with samples ξ has an unbiased stochastic gradient with bounded variance σ^2, i.e.,*

$$\mathbb{E}[\nabla_z\mathcal{L}(z;\xi)] = \nabla_z\mathcal{L}(z) \tag{10}$$

$$\mathbb{E}\|\nabla_z\mathcal{L}(z;\xi) - \nabla_z\mathcal{L}(z)\|^2 \le \sigma^2 \tag{11}$$

Assumption 2. *(Gradient Lipschitz) The loss function $\mathcal{L}(\mathcal{M})$ has a L-Lipschitz gradient, i.e., for $\forall z_1, z_2$ two different network weights, we have*

$$\|\nabla_z\mathcal{L}(z_1) - \nabla_z\mathcal{L}(z_2)\| \le L\|z_1 - z_2\| \tag{12}$$

Assumption 1 and 2 are standard assumptions in stochastic optimization and are widely used in deep learning convergence analysis [1, 2, 16, 45, 46, 56, 58].

Lemma 1. *(Lemma 1 in [16]) Assume $g(z)$ is a convex and possibly nonsmooth function. Let $z^+_{t+1} = \arg\min_z \big\{\langle z, \nabla_z\mathcal{L}(z)\rangle + \frac{1}{\eta}D_{\phi_t}(z, z_t) + g(z)\big\}$ and $\mathcal{P}_t = \frac{1}{\eta}(z_t - z^+_{t+1})$. Then we have*

$$\langle \nabla_z\mathcal{L}(z), \mathcal{P}_t\rangle \ge \epsilon\|\mathcal{P}_t\|^2 + \frac{1}{\eta}\big[g(z^+_{t+1}) - g(z_t)\big] \tag{13}$$

where $\epsilon > 0$ depends on ϵ-strongly convex function $\phi_t(z)$.

Based on Lemma 1, let $z_{t+1} = \arg\min_z \big\{\langle z, m_t\rangle + \frac{1}{\eta}D_{\phi_t}(z, z_t) + g(z)\big\}$, and define $\tilde{\mathcal{P}}_t = \frac{1}{\eta}[z_t - z_{t+1}]$. We have

$$\langle m_t, \tilde{\mathcal{P}}_t\rangle \ge \epsilon\|\tilde{\mathcal{P}}_t\|^2 + \frac{1}{\eta}(g(z_{t+1}) - g(x_t)) \tag{14}$$

Lemma 2. *Assume that the stochastic partial derivatives m_{t+1} be generated from Algorithm 1, we have*

$$\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - m_{t+1}\|^2 \le (1 - \alpha_{t+1})\mathbb{E}\|\nabla_z\mathcal{L}(z_t)) - m_t\|^2 + \frac{L^2\eta_t^2}{\alpha_{t+1}}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\alpha_{t+1}^2\sigma^2}{b} \tag{15}$$

Algorithm 3 ATO Algorithm (Mirror Descant)

```
Input: Target model with model weights vector $z$ (no need to be pre-tained). Datasets $D$, $D_{\text{CN}}$, $\gamma$, $\lambda$, $\eta$, total training iteration $T$ and each epoch of $\mathcal{D}$ has $Q$ iterations; controller network training steps $T_{\text{start}}$ and $T_{\text{end}}$.
for $t = 1, 2 \dots, T$ do
    Sample $\xi = (x, y)$ and compute the stochastic gradient $\nabla \mathcal{L}(z; \xi)$
    Compute gradient estimator $m_t = (1 - \alpha_t)m_{t-1} + \alpha_t \nabla \mathcal{L}(z; \xi)$
    Update model weight $z_{t+1} = \arg\min_z \left\{\langle m_t, z\rangle + \frac{1}{\eta_t} D_{\phi_t}(z, z_t) + g(z)\right\}$;
    if $T_{\text{start}}Q \leq T \leq T_{\text{end}}Q$ then
        $\mathcal{W}, \mathbf{w} \leftarrow$ CN-Update$(\mathcal{M}, \mathcal{W}, \mathbf{w}, D_{\text{C}})$
    end if
end for
Output: Directly remove pruned structures and construct a slimmer model.
```

Proof. Since $m_{t+1} = \alpha_{t+1}\nabla_z \mathcal{L}(z_{t+1}; \xi_{t+1}) + (1 - \alpha_{t+1})m_t$, we have

$$\begin{aligned}
&\mathbb{E}\|\nabla_z \mathcal{L}(z_{t+1}) - m_{t+1}\|^2 = \mathbb{E}\|\nabla_z \mathcal{L}(z_{t+1}) - \alpha_{t+1}\nabla_z \mathcal{L}(z_{t+1}; \xi_{t+1}) - (1 - \alpha_{t+1})m_t\|^2 \\
=&\mathbb{E}\|\alpha_{t+1}(\nabla_z \mathcal{L}(z_{t+1}) - \nabla_z \mathcal{L}(z_{t+1}; \xi_{t+1})) + (1 - \alpha_{t+1})(\nabla_z \mathcal{L}(z_t) - m_t) \\
&\quad + (1 - \alpha_{t+1})\big(\nabla_z \mathcal{L}(z_{t+1}) - \nabla_z \mathcal{L}(z_t)\big)\|^2 \\
\overset{(a)}{=}&\mathbb{E}\|(1 - \alpha_{t+1})(\nabla_z \mathcal{L}(z_t) - m_t) + (1 - \alpha_{t+1})\big(\nabla_z \mathcal{L}(z_{t+1}) - \nabla_z \mathcal{L}(z_t)\big)\|^2 \\
&\quad + \alpha_{t+1}^2 \mathbb{E}\|\nabla_z \mathcal{L}(z_{t+1}) - \nabla_z \mathcal{L}(z_{t+1}; \xi_{t+1})\|^2 \\
\leq&(1 - \alpha_{t+1})^2(1 + \frac{1}{\alpha_{t+1}})\mathbb{E}\|\nabla_z \mathcal{L}(z_{t+1}) - \nabla_z \mathcal{L}(z_t)\|^2 \\
&\quad + (1 - \alpha_{t+1})^2(1 + \alpha_{t+1})\mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 + \alpha_{t+1}^2 \mathbb{E}\|\nabla_z \mathcal{L}(z_{t+1}) - \nabla_z \mathcal{L}(z_{t+1}; \xi_{t+1})\|^2 \\
\overset{(b)}{\leq}&(1 - \alpha_{t+1})\mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 + \frac{1}{\alpha_{t+1}}\mathbb{E}\|\nabla_z \mathcal{L}(z_{t+1}) - \nabla_z \mathcal{L}(z_t))\|^2 + \frac{\alpha_{t+1}^2 \sigma^2}{b} \\
\leq&(1 - \alpha_{t+1})\mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 + \frac{L^2}{\alpha_{t+1}}\mathbb{E}\|z_{t+1} - z_t\|^2 + \frac{\alpha_{t+1}^2 \sigma^2}{b} \\
=&(1 - \alpha_{t+1})\mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 + \frac{L^2 \eta_t^2}{\alpha_{t+1}}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\alpha_{t+1}^2 \sigma^2}{b}
\end{aligned}$$

where the (a) is due to $\mathbb{E}_{\xi_{t+1}}[\nabla \mathcal{L}(z_{t+1}; \xi_{t+1})] = \nabla \mathcal{L}(z_{t+1})$; the (b) holds by $0 \leq \alpha_{t+1} \leq 1$ such that $(1 - \alpha_{t+1})^2(1 + \alpha_{t+1}) = 1 - \alpha_{t+1} - \alpha_{t+1}^2 + \alpha_{t+1}^3 \leq 1 - \alpha_{t+1}$ and $(1 - \alpha_{t+1})^2(1 + \frac{1}{\alpha_{t+1}}) \leq (1 - \alpha_{t+1})(1 + \frac{1}{\alpha_{t+1}}) = -\alpha_{t+1} + \frac{1}{\alpha_{t+1}} \leq \frac{1}{\alpha_{t+1}}$, and the b is the batch size; the last inequality holds by the definition of $\tilde{\mathcal{P}}_t$ in Eq. (14) □

Lemma 3. *Let* $z_{t+1} = \arg\min_z \left\{\langle m_t, z\rangle + \frac{1}{\eta_t} D_{\phi_t}(z, z_t) + g(z)\right\}$ *and* $z_{t+1}^{+} = \arg\min_z \{\langle \nabla \mathcal{L}(z_t), z\rangle + \frac{1}{\eta_t} D_{\phi_t}(z, z_t) + g(z)\Big\}$, *we have*

$$\|\nabla \mathcal{L}(z_t) - m_t\| \geq \epsilon \left\|\mathcal{P}_t - \tilde{\mathcal{P}}_t\right\| \tag{16}$$

Proof. based on the definition of z_{t+1} and z_t, and the convex property, we have,

$$\left\langle m_t + \nabla g(z_{t+1}) + \frac{1}{\eta_t}\left(\nabla \mathcal{L}_t(z_{t+1}) - \nabla \mathcal{L}_t(z_t)\right), z - z_{t+1}\right\rangle \geq 0 \tag{17}$$

$$\left\langle \nabla \mathcal{L}(z_t) + \nabla g\left(z_{t+1}^{+}\right) + \frac{1}{\eta_t}\left(\nabla \mathcal{L}_t\left(z_{t+1}^{+}\right) - \nabla \mathcal{L}_t(z_t)\right), z - z_{t+1}^{+}\right\rangle \geq 0 \tag{18}$$

where $\nabla g(z_{t+1}) \in \partial g(z_{t+1})$. Taking $z = z_{t+1}^{+}$in the Eq. (17) and $z = z_{t+1}$ in the Eq. (18), by the convexity of $g(x)$, we have

$$\begin{aligned}\left\langle m_t, z_{t+1}^+ - z_{t+1}\right\rangle &\geq \left\langle \nabla g\left(z_{t+1}\right), z_{t+1} - z_{t+1}^+\right\rangle + \frac{1}{\eta_t}\left\langle \nabla\mathcal{L}_t\left(z_{t+1}\right) - \nabla\mathcal{L}_t\left(z_t\right), z_{t+1} - z_{t+1}^+\right\rangle \\ &\geq g\left(z_{t+1}\right) - g\left(z_{t+1}^+\right) + \frac{1}{\eta_t}\left\langle \nabla\mathcal{L}_t\left(z_{t+1}\right) - \nabla\mathcal{L}_t\left(z_t\right), z_{t+1} - z_{t+1}^+\right\rangle \end{aligned} \tag{19}$$

$$\begin{aligned}\left\langle \nabla\mathcal{L}\left(z_t\right), z_{t+1} - z_{t+1}^+\right\rangle &\geq \left\langle \nabla g\left(z_{t+1}^+\right), z_{t+1}^+ - z_{t+1}\right\rangle + \frac{1}{\eta_t}\left\langle \nabla\mathcal{L}_t\left(z_{t+1}^+\right) - \nabla\mathcal{L}_t\left(z_t\right), z_{t+1}^+ - z_{t+1}\right\rangle \\ &\geq g\left(z_{t+1}^+\right) - g\left(z_{t+1}\right) + \frac{1}{\eta_t}\left\langle \nabla\mathcal{L}_t\left(z_{t+1}^+\right) - \nabla\mathcal{L}_t\left(z_t\right), z_{t+1}^+ - z_{t+1}\right\rangle \end{aligned} \tag{20}$$

Summing up the above inequalities, we obtain

$$\left\langle \nabla\mathcal{L}\left(z_t\right) - m_t, z_{t+1} - z_{t+1}^+\right\rangle \geq \frac{1}{\eta_t}\left\langle \nabla\mathcal{L}_t\left(z_{t+1}^+\right) - \nabla\mathcal{L}_t\left(z_{t+1}\right), z_{t+1}^+ - z_{t+1}\right\rangle \geq \frac{\epsilon}{\eta_t}\left\|z_{t+1}^+ - z_{t+1}\right\|^2$$

where the last inequality is due to the ϵ-strongly convex function $\mathcal{L}_t(z)$. Since $\left\|\nabla\mathcal{L}\left(z_t\right) - m_t\right\|\left\|z_{t+1} - z_{t+1}^+\right\| \geq \left\langle \nabla\mathcal{L}\left(z_t\right) - m_t, z_{t+1} - z_{t+1}^+\right\rangle$ and $\left\|\mathcal{P}_t - \tilde{\mathcal{P}}_t\right\| = \|\frac{1}{\eta_t}\left(z_t - z_{t+1}^+\right) - \frac{1}{\eta_t}\left(z_t - z_{t+1}\right)\Big\| = \frac{1}{\eta_t}\Big\| z_{t+1}^+ - z_{t+1}\|$, we have

$$\left\|\nabla\mathcal{L}\left(z_t\right) - m_t\right\| \geq \epsilon\left\|\mathcal{P}_t - \tilde{\mathcal{P}}_t\right\| \tag{21}$$

□

Lemma 4. *Suppose the sequence $\{z_t\}_{t=1}^T$ be generated from Algorithms 1. Let $0 < \eta_t \leq \min\{1, \frac{\epsilon}{4L}\}$, we have*

$$\mathcal{J}\left(z_{t+1}\right) \leq \mathcal{J}\left(z_t\right) - \frac{\eta_t\epsilon}{4}\left\|\mathcal{P}_t\right\|^2 + \frac{2\eta_t}{\epsilon}\left\|m_t - \nabla_z\mathcal{L}\left(z_t\right)\right\|^2 \tag{22}$$

Proof. Since $z_{t+1} = \arg\min_z\left\{\langle m_t, z\rangle + \frac{1}{\eta_t}D_{\phi_t}\left(z, z_t\right) + g(z)\right\}$ and $\tilde{\mathcal{P}}_t = \frac{1}{\eta}(z_t - z_{t+1})$, and function $\mathcal{L}(z)$ has L-Lipschitz continuous gradient. we have

$$\begin{aligned}\mathcal{L}\left(z_{t+1}\right) &\leq \mathcal{L}\left(z_t\right) + \left\langle \nabla\mathcal{L}\left(z_t\right), z_{t+1} - z_t\right\rangle + \frac{L}{2}\left\|z_{t+1} - z_t\right\|^2 \\ &= \mathcal{L}\left(z_t\right) - \eta_t\left\langle \nabla\mathcal{L}\left(z_t\right), \tilde{\mathcal{P}}_t\right\rangle + \frac{\eta_t^2 L}{2}\left\|\tilde{\mathcal{P}}_t\right\|^2 \\ &= \mathcal{L}\left(z_t\right) - \eta_t\left\langle m_t, \tilde{\mathcal{P}}t\right\rangle + \eta_t\left\langle m_t - \nabla\mathcal{L}\left(z_t\right), \tilde{\mathcal{P}}_t\right\rangle + \frac{\eta_t^2 L}{2}\left\|\tilde{\mathcal{P}}_t\right\|^2 \\ &\overset{(a)}{\leq} \mathcal{L}\left(z_t\right) - \eta_t\epsilon\left\|\tilde{\mathcal{P}}_t\right\|^2 - g\left(z_{t+1}\right) + g\left(z_t\right) + \eta_t\left\langle m_t - \nabla\mathcal{L}\left(z_t\right), \tilde{\mathcal{P}}_t\right\rangle + \frac{\eta_t^2 L}{2}\left\|\tilde{\mathcal{P}}_t\right\|^2 \\ &\overset{(b)}{\leq} \mathcal{L}\left(z_t\right) + \left(\frac{\eta_t^2 L}{2} - \frac{3\eta_t\epsilon}{4}\right)\left\|\tilde{\mathcal{P}}_t\right\|^2 - g\left(z_{t+1}\right) + g\left(z_t\right) + \frac{\eta_t}{\epsilon}\left\|m_t - \nabla\mathcal{L}\left(z_t\right)\right\|^2 \end{aligned} \tag{23}$$

where the (a)holds by the above Lemma 1, and the (b) holds by the following Cauchy-Schwarz inequality and Young's inequality as

$$\begin{aligned}\left\langle m_t - \nabla\mathcal{L}\left(z_t\right), \tilde{\mathcal{P}}_t\right\rangle &\leq \left\|m_t - \nabla\mathcal{L}\left(z_t\right)\right\|\left\|\tilde{\mathcal{P}}_t\right\| \\ &\leq \frac{1}{\epsilon}\left\|m_t - \nabla\mathcal{L}\left(z_t\right)\right\|^2 + \frac{\epsilon}{4}\left\|\tilde{\mathcal{P}}_t\right\|^2 \end{aligned}$$

Since $\mathcal{J}(z) = \mathcal{L}(z) + g(z)$, we have

$$\begin{aligned}\mathcal{J}\left(z_{t+1}\right) &\leq \mathcal{J}\left(z_t\right) + \left(\frac{\eta_t^2 L}{2} - \frac{3\eta_t\epsilon}{4}\right)\left\|\tilde{\mathcal{P}}_t\right\|^2 + \frac{\eta_t}{\epsilon}\left\|m_t - \nabla_z\mathcal{L}\left(z_t\right)\right\|^2 \\ &\leq \mathcal{J}\left(z_t\right) - \frac{5\eta_t\epsilon}{8}\left\|\tilde{\mathcal{P}}_t\right\|^2 + \frac{\eta_t}{\epsilon}\left\|m_t - \nabla_z\mathcal{L}\left(z_t\right)\right\|^2 \end{aligned} \tag{24}$$

where the last inequality is due to $0 < \eta_t \le \frac{\epsilon}{4L}$. Based on Lemma 3, we have

$$\|\mathcal{P}_t\|^2 \le 2\left\|\tilde{\mathcal{P}}_t\right\|^2 + 2\left\|\tilde{\mathcal{P}}_t - \mathcal{P}_t\right\|^2 \le 2\left\|\tilde{\mathcal{P}}_t\right\|^2 + \frac{2}{\epsilon^2}\left\|m_t - \nabla\mathcal{L}\left(z_t\right)\right\|^2 \tag{25}$$

Finally, we have

$$\mathcal{J}\left(z_{t+1}\right) \le \mathcal{J}\left(z_t\right) - \frac{\eta_t\epsilon}{8}\left\|\tilde{\mathcal{P}}_t\right\|^2 - \frac{\eta_t\epsilon}{4}\|\mathcal{P}_t\|^2 + \frac{2\eta_t}{\epsilon}\left\|m_t - \nabla_z\mathcal{L}\left(z_t\right)\right\|^2 \tag{26}$$

□

Theorem 2. *(Restatement of Theorem 1) Assume that the sequence $\{z_t\}_{t=1}^{T}$ be generated from the Algorithm 1. When we have $\eta_t = \frac{\hat{c}}{(\bar{c}+t)^{1/2}}$, $\frac{\hat{c}}{\bar{c}^{1/2}} \le \min\{1, \frac{\epsilon}{4L}\}$, $c_1 = \frac{4L}{\epsilon}$, $\alpha_{t+1} = c_1\eta_t$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathcal{P}_t\| \le \frac{\sqrt{G}\bar{c}^{1/4}}{T^{1/2}} + \frac{\sqrt{G}}{T^{1/4}} \tag{27}$$

where $G = \frac{4(\mathcal{J}(z_1)-\mathcal{J}(z^*))}{\epsilon\hat{c}} + \frac{2\sigma^2}{bL\epsilon\hat{c}} + \frac{2\bar{c}\sigma^2}{\hat{c}\epsilon Lb}\ln(\bar{c}+T)$.

Proof. $\eta_t = \frac{\hat{c}}{(\bar{c}+t)^{1/2}}$ on t is decreasing, $\eta_t \le \eta_0 = \frac{\hat{c}}{\bar{c}^{1/2}} \le \min\{1, \frac{\epsilon}{4L}\}$ for any $t \ge 0$. At the same time, $c_1 = \frac{4L}{\epsilon}$. We have $\alpha_{t+1} = c_1\eta_t \le \frac{c_1\hat{c}}{\bar{c}^{1/2}} \le 1$.

According to Lemma 2, we have

$$\begin{aligned}
&\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - m_{t+1}\|^2 - \mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 \\
\le& -\alpha_{t+1}\mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 + L^2\eta_t^2/\alpha_{t+1}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\alpha_{t+1}^2\sigma^2}{b} \\
=& -c_1\eta_t\mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 + L^2\eta_t/c_1\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{c_1^2\eta_t^2\sigma^2}{b} \\
\le& -\frac{4L\eta_t}{\epsilon}\mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 + \frac{\epsilon L\eta_t}{4}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\bar{c}\eta_t^2\sigma^2}{\hat{c}^2 q}
\end{aligned} \tag{28}$$

where the above equality holds by $\alpha_{t+1} = c_1\eta_t$, and the last inequality is due to $c_1 = \frac{4L}{\epsilon}$.

According to Lemma 4, we have

$$\mathcal{J}\left(z_{t+1}\right) \le \mathcal{J}\left(z_t\right) - \frac{\eta_t\epsilon}{8}\left\|\tilde{\mathcal{P}}_t\right\|^2 - \frac{\eta_t\epsilon}{4}\|\mathcal{P}_t\|^2 + \frac{2\eta_t}{\epsilon}\left\|m_t - \nabla_z\mathcal{L}\left(z_t\right)\right\|^2 \tag{29}$$

Next, we define a *Lyapunov* function, for any $t \ge 1$

$$\Omega_t = \mathbb{E}\big[\mathcal{J}(z_t) + \frac{1}{2L}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2\big]$$

Then we have

$$\begin{aligned}
&\Omega_{t+1} - \Omega_t = \mathbb{E}\big[\mathcal{J}(z_{t+1}) - \mathcal{J}(z_t)\big] + \frac{1}{2}[\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - m_{t+1}\|^2 - \mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2] \\
\le& -\frac{\eta_t\epsilon}{8}\left\|\tilde{\mathcal{P}}_t\right\|^2 - \frac{\eta_t\epsilon}{4}\|\mathcal{P}_t\|^2 + \frac{2\eta_t}{\epsilon}\left\|m_t - \nabla_z\mathcal{L}\left(z_t\right)\right\|^2 + \frac{1}{2L}\left(-\frac{4L\eta_t}{\epsilon}\mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 + \frac{\epsilon L\eta_t}{4}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\bar{c}\eta_t^2\sigma^2}{\hat{c}^2 q}\right) \\
\le& -\frac{\epsilon\eta_t}{4}\mathbb{E}\|\mathcal{P}_t\|^2 + \frac{\bar{c}\sigma^2}{2\hat{c}^2 Lq}\eta_t^2
\end{aligned}$$

Then we have

$$\eta_t\mathbb{E}\|\mathcal{P}_t\|^2 \le \frac{4(\Omega_t - \Omega_{t+1})}{\epsilon} + \frac{2\bar{c}\sigma^2}{\epsilon\hat{c}^2 Lb}\eta_t^2 \tag{30}$$

Taking average over $t = 1, 2, \cdots, T$ on both sides of (30), we have

$$\frac{1}{T}\sum_{t=1}^{T}\eta_t\mathbb{E}\|\mathcal{P}_t\|^2 \leq \sum_{t=1}^{T}\frac{4(\Omega_t - \Omega_{t+1})}{T\epsilon} + \frac{1}{T}\sum_{t=1}^{T}\frac{2\bar{c}\sigma^2}{\epsilon L\hat{c}^2 b}\eta_t^2. \tag{31}$$

In addition, we have

$$\Omega_1 = \mathcal{J}(z_1) + \frac{1}{2L}\mathbb{E}\|\nabla_z\mathcal{L}(z_1) - v_1\|^2 \leq \mathcal{J}(z_1) + \frac{\sigma^2}{2bL}, \tag{32}$$

where the above inequality holds by Assumption 1. Since η_t is decreasing on t, i.e., $\eta_T^{-1} \geq \eta_t^{-1}$ for any $0 \leq t \leq T$, we have

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathcal{P}_t\|^2 &\leq \frac{4}{T\epsilon\eta_T}\sum_{t=1}^{T}\big(\Omega_t - \Omega_{t+1}\big) + \frac{1}{T\eta_T}\sum_{t=1}^{T}\frac{2\bar{c}\sigma^2}{\epsilon L\hat{c}^2 b}\eta_t^2 \\
&\leq \frac{4}{T\epsilon\eta_T}\big(\mathcal{J}(z_1) + \frac{\sigma^2}{2bL} - \mathcal{J}(z^*)\big) + \frac{1}{T\eta_T}\sum_{t=1}^{T}\frac{2\bar{c}\sigma^2}{\epsilon L\hat{c}^2 b}\eta_t^2 \\
&\leq \frac{4(\mathcal{J}(z_1) - \mathcal{J}(z^*))}{T\epsilon\eta_T} + \frac{2\sigma^2}{bL\epsilon\eta_T T} + \frac{2\bar{c}\sigma^2}{\eta_T T\epsilon L\hat{c}^2 b}\int_1^T\frac{\hat{c}^2}{\bar{c}+t}dt \\
&\leq \frac{4(\mathcal{J}(z_1) - \mathcal{J}(z^*))}{T\epsilon\eta_T} + \frac{2\sigma^2}{bL\epsilon\eta_T T} + \frac{2\bar{c}\sigma^2}{\eta_T T\epsilon Lb}\ln(\bar{c}+T) \\
&= \left(\frac{4(\mathcal{J}(z_1) - \mathcal{J}(z^*))}{\epsilon\hat{c}} + \frac{2\sigma^2}{bL\epsilon\hat{c}} + \frac{2\bar{c}\sigma^2}{\hat{c}\epsilon Lb}\ln(\bar{c}+T)\right)\frac{(\bar{c}+T)^{1/2}}{T},
\end{aligned} \tag{33}$$

where the second inequality holds by the above inequality (32) and the fact that $\mathcal{J}_{T+1} \geq \mathcal{J}^*$ Let $G = \frac{4(\mathcal{J}(z_1)-\mathcal{J}(z^*))}{\epsilon\hat{c}} + \frac{2\sigma^2}{bL\epsilon\hat{c}} + \frac{2\bar{c}\sigma^2}{\hat{c}\epsilon Lb}\ln(\bar{c}+T)$, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[|\mathcal{P}_t\|^2\big] \leq \frac{G}{T}(\bar{c}+T)^{1/2}. \tag{34}$$

According to Jensen's inequality, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\mathcal{P}_t\|\big] \leq \left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\mathcal{P}_t\|^2\big]\right)^{1/2} \leq \frac{\sqrt{G}}{T^{1/2}}(\bar{c}+T)^{1/4} \leq \frac{\sqrt{G}\bar{c}^{1/4}}{T^{1/2}} + \frac{\sqrt{G}}{T^{1/4}} \tag{35}$$

where the last inequality is due to $(a+b)^{1/4} \leq a^{1/4} + b^{1/4}$ for all $a, b > 0$. Thus, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathcal{P}_t\| \leq \frac{\sqrt{G}\bar{c}^{1/4}}{T^{1/2}} + \frac{\sqrt{G}}{T^{1/4}} \tag{36}$$

□