Convex and Non-convex Model Compression for Large-Scale Model Training

by

Xidong Wu

M.S., University of California, Berkeley, 2020

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2024

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Xidong Wu

It was defended on

April 1 2024

and approved by

Yufei Huang, PhD, Professor, Department of Electrical and Computer Engineering, Department of Pharmaceutical Sciences, School of Medicine

Ahmed Dallal, PhD, Assistant Professor, Department of Electrical and Computer Engineering

Liang Zhan, PhD, Professor, Department of Electrical and Computer Engineering

Bo Zeng, PhD, Associate Professor, Department of Industrial Engineering, Department of Electrical and Computer Engineering

Mingui Sun, PhD, Professor, Department of Neurological Surgery, School of Medicine

Dissertation Director: Zhi-Hong Mao, PhD, Professor, Department of Electrical and Computer Engineering

Convex and Non-convex Model Compression for Large-Scale Model Training

Xidong Wu, PhD

University of Pittsburgh, 2024

Recently, machine learning (ML) models have gained extensive utilization across a variety of applications. However, unlike traditional methods, ML models, especially those employing deep learning, utilize their significant depth and intricate architecture to improve approximation capabilities, posing a challenge to the hardware capabilities of devices during deployment and communication during training. This challenge is particularly pronounced when deploying ML models on edge devices, which are characterized by limited storage and modest processing capabilities. To address these issues, the concept of model compression has emerged as an approach to reduce the size of ML models with minimal performance degradation and facilitate deployment. Several model compression techniques, such as weight pruning, knowledge distillation, and model screening, have been explored. Additionally, the training process for these models requires substantial data, and distributed/federated training serves as a solution to overcome data-related obstacles. The objective of this dissertation is to enhance the efficiency of convex and non-convex models, with or without multi-party collaborative training (distributed and federated learning).

We develop various approaches for compressing logistic classifiers (convex models) and deep learning models (nonconvex models). In task 1, we introduce a novel distributed dynamic safe screening framework for generalized sparse convex models. This framework reduces the model dimension in advance compared to traditional lasso techniques, accelerating distributed training and reducing communication overhead by discarding inactive features with zero coefficients. In task 2, we concentrate on the application of foundation models in federated learning. Foundation models exhibit outstanding performance and the ability to mitigate the impact of heterogeneous data distributions. We explore compressing foundation models to improve performance on edge devices. In task 3, we delve into structural pruning in centralized learning. We propose a new algorithm that employs the controller network to guide end-to-end model pruning without relying on additional fine-tuning procedures af-

ter removing redundant structures. Comprehensive experiments conducted on a large scale within distributed or centralized settings validate the rationale and efficacy of our proposed methodology. Finally, we provide related theoretical analysis to ensure the convergence of proposed algorithms.

Table of Contents

List of Tables

List of Figures

Preface

In writing this thesis, my primary goal was to explore convex and non-convex model compression in ML training. Throughout this journey, I encountered many challenges but was fortunate to receive guidance and support from my advisors, Prof. Mao and Prof. Huang. I also appreciate committee members, Prof. Ahmed Dallal, Prof. Liang Zhan, Prof. Bo Zeng, and Prof. Mingui Sun.

1.0 Introduction

1.1 Motivation and Research Objective

Machine learning (ML) has achieved notable accomplishments across various tasks [10, 38], such as computer vision, speed recognition, and natural language processing. To address real-world complexities, contemporary ML models are progressively expanding in terms of feature dimensions, width, and depth, from convex to nonconvex, and other aspects. Such augmented capacities enable ML models to attain superior performance on diverse benchmarks, albeit at the expense of heightened computational and storage demands.

Simultaneously, as edge devices, such as mobile and embedded devices, undergo recent advancements, the desire to deploy ML models on these platforms has surged significantly. Nevertheless, an inherent conflict emerges between the dimensions of ML models and the hardware capabilities of such edge devices. At the same time, large-scale ML models put a huge communication burden across different clients while usually more than one clients are involved in the ML model training. To surmount these challenges, numerous endeavors [5, 36, 37] have endeavored to minimize the size of ML models, rendering them amenable for deployment on embedded or mobile devices.

Numerous techniques exist for model compression. Notably, the techniques of model screening for convex models, and model distillation and model pruning for deep learning models have gained prominence. In this dissertation, we will focus on the applications of model screening, model distillation, and structural pruning. At the same, our dissertation is centered on investigating these methodologies for model training in the case with or without multi-party collaborative training (distributed and federated learning).

Sparse representations have a significant role in various ML applications within the realm of convex (non-deep) ML models [79, 18, 9, 112]. Over the preceding decades, numerous models incorporating sparse regularization have achieved remarkable triumphs in scenarios characterized by high dimensionality, owing to their capability to promote model sparsity [111, 4]. The lasso model is a classical way to get a sparse convex model with l-1 norm.

Within Nonconvex (deep) models, pruning is another way of model compression technique to get sparse representations, which involves removing certain parts of an ML model, typically weight or channel, to reduce its size and computational complexity. There are two main types of pruning techniques: weight pruning and structural pruning. Compared with weight pruning, structural pruning [30, 31], particularly channel pruning, is more hardware-friendly since we can control the FLOAPs reduction to achieve computational and storage savings.

Model distillation [43] emerges as another prevalent technique for compressing models as the development of foundation models. The fundamental idea behind knowledge distillation involves acquiring a distilled model, or student network, and then training this more concise neural network to replicate the outputs generated by a larger network, known as the teacher network. In the training process of the student network, it can draw upon not just the logits of the teacher network, but also leverage the intermediate outcomes. Research of foundation models boosts the application of model distillation.

Model compression is also widely used in multi-party collaborative training. Achieving effective generalization of deep models on unseen test data often necessitates a substantial volume of training data. Direct collating data from all available establishments or clients elicits concerns about data privacy. For example, the act of accumulating medical data is a resource-intensive endeavor, as previously highlighted, and these data have evolved into a valuable asset for medical institutions. However, medical institutions carry the responsibility of safeguarding data collected from patients. The process of data aggregation may consequently expose patients to the potential jeopardy of data breaches. Under these circumstances, the adoption of distributed learning and federated learning becomes imperative. However, the large size of ML models put a heavy commutation overhead during training. Model compression is a good way to reduce model size, save commutation overhead, and speed up training.

Although federated learning [56] stands out for its efficiency in communication since maybe a part of the clients participate in global commutation in each round after multiple local training compared with distributed learning, cross-client federated learning is highly related to the edge device. A notable issue in federated learning is client drift. This arises due to the non-iid (non-independent and identically distributed) data distribution, leading

to divergent trajectories of client models in the absence of timely updates to the global models through communication. Although the application of foundation models in federated learning could mitigate client drift issues, it is impossible to deploy foundation models in edge devices, such as smartphones or embedded devices. Overall, model compression has a wide application in the ML model training, including non-deep (convex) models or deep (convex) models, centralized training, or multi-party collaborative training (distributed learning or federated learning).

Overall, the research objective of this dissertation is convex and non-convex model compression, including model screening, model distillation, and structural pruning. We also consider distributed training since it is more complicated than centralized training.

1.2 Main Contributions

We study the model compression in the ML models training, including non-deep (convex) models or deep (convex) models, and centralized training or multi-party collaborative training (distributed learning or federated learning). There are three parts we completed as below:

First, we consider the model compression with sparse representation in the convex models, such as logistic regression, and we consider the model training in a distributed setting since it is more complicated than centralized learning. We present an innovative distributed dynamic safe screening framework designed for generalized sparse convex models. This framework dynamically diminishes the model's dimensionality in advance compared to conventional lasso techniques, thereby expediting distributed training and minimizing communication overhead through the elimination of inactive features with zero coefficients.

Afterward, we focus on the deep (non-convex) models since deep learning presents remarkable performance. In the second part, we explore the sparse representation with structural pruning for centralized learning. In the course of this manuscript, we use the term "centralized training" to describe the process of collecting data from clients and subsequently training the model without considering the communication. It is important to acknowledge

that centralized training presents an upper-performance limit for multi-party collaborative training algorithms. 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. It is important to highlight that numerous existing pruning methods often exhibit significant limitations, necessitating a complex multi-stage process. This process usually involves initial pre-training, intermediate training to identify redundancies and subsequent fine-tuning. Effectively managing this multi-stage process in ML training requires considerable engineering efforts and specialized expertise. Following OTO [16] and OTOv2 [17], we advocate for end-to-end training and pruning frameworks with a controller network.

Finally, we focus on the model distillation for deep learning models in federated learning. We leverage foundation models to guide the lightweight model training in federated learning to improve inference efficiency. Similar to various distributed learning approaches, Federated Learning (FL) faces a significant challenge wherein client data stored locally exhibit heterogeneity, stemming from uneven distributions or unbalanced patterns [64]. While foundation models offer a potential solution to maintaining model performance under such heterogeneous data distributions, their application on edge devices is impractical. To effectively harness the performance benefits of foundation models under non-IID data distribution, while utilizing small-scale backbones for quicker inference, we investigate the approach of distilling knowledge from foundation models into the smaller client models. The experiments show the performance of the lightweight model experiences substantial improvement through the incorporation of strong prior knowledge derived from foundation models.

We summarize our contribution as follows.

- We propose a new distributed dynamic safe screening (DDSS) framework for generalized sparse (convex) models, which is easy to implement on both shared-memory and distributed-memory architecture. To the best of my knowledge, this is the first work of distributed dynamic safe screening. At the same time, we also consider the shared-memory parallel setting. We conduct lots of experiments to verify the efficiency of our algorithms. It could largely save the commutation overhead due to the reduction of model dimension and speed up model training. In addition, the experiments show the

linear speed-up of our algorithms as the number of clients.

- Then we introduce an innovative structural pruning algorithm, Auto-Train-Once (ATO), designed to automatically reduce the computational and storage costs of non-convex ML models. It is an end-to-end algorithm since it does not rely on extra fine-tuning steps to train from scratch and we only train the target model once. A controller network is used to dynamically generate the binary mask to guide the pruning of the target model. In addition, we present a theoretical analysis under mild assumptions to guarantee convergence, along with extensive experiments.
- Finally, we explore the application of foundation models in federated learning (Fed-LPFM). Preliminary results show the remarkable performance of foundation models after fine-tuning in heterogeneous distribution. In order to deploy the model in the edge device, we show that with the help of a foundation model, tiny models could overcome the client drift to some extent.

1.3 Orgnization of Dissertation

This dissertation is organized as below:

- Chapter 1 is the introduction, which includes the motivation, research objective, and main contribution.
- In Chapter 2, we present related works and some background for the following chapters. Distributed static screening, and asynchronous stochastic method parts are related to Task 1; the model pruning part is related to Task 2; and federated learning, foundation models, and model distillation parts are related to Task 3.
- Chapter 3 is Task 1 about present Dynamic Model Screening Algorithms for Distributed Training.
- Chapter 4 is Task 2 about Auto-Train-Once (ATO) to discuss how to use a controller network to guide automatic network pruning from scratch.
- Chapter 5 presents Task 3 about leveraging foundation models in efficient federated learning with model distillation.

- Chapter 6 is the conclusion of this dissertation.

2.0 Related Works

Here we introduce existing different approaches to improve the non-deep (convex) or deep (non-convex) model performance in the case with or without models with or without multi-party collaborative training (distributed and federated learning).

2.1 Distributed Static Screening

The study most closely related to ours is Parallel Static Screening (PSE) [62]. It addresses the challenges posed by high-dimensional settings through the implementation of the parallel elimination (PE) step preceding Asynchronous Grouped Coordinate Descent (AGCD). However, the proposed approach comes with several drawbacks. Firstly, it is performed only once before the optimization algorithm. Secondly, the application of the safe static rule [33] in PE is secure, whereas the use of the strong rule [97] in PE is deemed unsafe, potentially leading to erroneous feature discards. Thirdly, its determinism on samples hinders scalability for large n. Fourthly, PE-AGCD cannot leverage the sparsity of the data. Lastly, PE-AGCD is constrained to Lasso regression.

Table 1 presents a comparison between our DDSS method and PSE, highlighting the advantages of DDSS. Firstly, DDSS operates dynamically throughout the entire training process, providing a dual benefit of acceleration and leveraging the convergence of the optimization algorithm. Additionally, DDSS ensures the safety of the training process while maintaining model accuracy. Thirdly, DDSS adopts a stochastic approach, making it adaptable to both sample and feature scalability. Finally, DDSS capitalizes on data sparsity through the utilization of sparse proximal updates, further contributing to the acceleration of the training process.

2.2 Asynchronous Stochastic Method

In the work by [81], they introduced an asynchronous doubly stochastic approach, but it overlooks the consideration for sparsity, potentially leading to sluggish performance. Another advancement, presented in [35], provides an asynchronous doubly stochastic method specifically designed for group regularized learning problems. [58] also proposed an asynchronous sparse incremental gradient method; however, it falls short in capitalizing on the model's sparsity and lacks the adaptability to the dynamic screening method for efficient training. [45] mainly focuses on the asynchronous mini-batch gradient descent with variance reduction (AsySVRG) for non-convex optimization. In contrast, our DDSS method stands out by effectively leveraging both the model's sparsity and the dataset, offering a unique advantage over these previous approaches.

2.3 Model Pruning

To minimize storage and computational expenses, pruning methods aim to identify and eliminate redundant structures from the complete model. There are two main types of pruning techniques: weight pruning and structural pruning. In weight pruning, individual connections (weights) in the neural network are set to zero or removed based on certain criteria. This reduces the number of parameters in the model and, consequently, its size. Common methods for weight pruning consist of setting small-weight values to zero with a specific thresholding, removing a certain percentage of the smallest weights. Structural Pruning involves removing entire unit structural from the neural network, such as channels. This can be more aggressive than weight pruning, as entire features or representations are discarded. It is often based on the impact on the model's output or activation values. Less influential structural are pruned to reduce the model's complexity.

Since pruning may lead to a significant loss of accuracy, fine-tuning and careful selection of pruning criteria are crucial to maintaining or even improving model performance during model compression. The majority of existing structural pruning techniques follow a

three-stage process: (1) train a full model from scratch; (2) pinpoint redundant structures based on various criteria; (3) fine-tune or retrain the pruned model to restore performance. Different methods make distinct choices regarding the pruning criteria. For instance, filter pruning, as introduced by [60], selects crucial structures with larger norm values. Beyond evaluating channel or filter importance solely based on magnitude, the scaling factor of batch normalization [46], a popular component in modern CNN architectures [38, 89], can also be employed to identify significant channels. [77] utilize sparse regularization on the scaling factors of batch normalization to facilitate channel pruning.

A channel undergoes pruning when its associated scaling factor is considered small. The concept of structure sparse selection [44] expands upon the utilization of scaling factors in batch normalization by applying it to diverse structures such as neurons, groups, or residual blocks. Sparsity regularization is additionally enforced on these structures. Another avenue of research [54, 107, 31, 50] involves the pruning of insignificant channels using learnable scaling factors assigned to each structure. These learnable parameters are designed to be differentiable end-to-end, allowing for gradient-based optimizations. Inter-channel dependency [86] is another approach to channel removal. Greedy forward selection [106] systematically adds channels with large norms to an initially empty model through iterative steps.

Furthermore, reinforcement learning and evolutionary algorithms offer alternative approaches for selecting important structures. The Automatic Model Compression (AMC) technique [42] employs a policy network to determine the width of each layer, with updates performed through policy gradient methods. In MetaPruning [76, 68], evolutionary algorithms are employed to discover the optimal combination of structures, and a hypernet is utilized to generate the model weights.

Conventional structural pruning methods typically necessitate manual intervention across all three stages, particularly in the second and third phases. To streamline and reduce manual efforts in all three stages, OTO [16] formulates a structured-sparsity optimization problem and introduces the Half-Space Stochastic Projected Gradient (HSPG) method to address it. OTO overcomes various drawbacks associated with prior methods based on structural sparsity [98, 67], such as (1) the involvement of multiple training stages due to the inability of their group partition to isolate the impact of pruned structures on the model

output, and (2) the need for heuristic post-processing to generate zero groups. Building upon OTO, OTOv2 [17] introduces automated Zero-Invariant Groups (ZIGs) partitioning and employs the Dual Half-Space Projected Gradient (DHSPG) method for enhanced user-friendliness and performance. Despite the advantages of OTOv2, it still exhibits several issues, including (1) the static selection of pruning groups in ZIGs, which cannot be altered with model weight updates; (2) the increased complexity of HSPG/DHSPG compared to simple proximal gradient operators; and (3) insufficient theoretical analysis with overly strong assumptions in both OTO and OTOv2. In this dissertation, we present solutions to address all the aforementioned weaknesses of OTOv2.

2.4 Federated Learning

FedAvg was proposed in [80] With periodic model averaging, it can dramatically reduce communication overheads. In FedAvg, clients will receive the starting model w_r from the server at training round r. Each client k performs E epochs of local training to update the local model to $w_{r+1}^{(k)}$ with the popular momentum SGD or Adam [55] optimizer depending on the application needs. Then the server gathers and averages the local models to $w_{r+1} = \frac{1}{K}\sum_{k=1}^{K} p_k w_{r+1}^{(k)}$.

Previous studies investigated federated learning (FL) algorithms within the context of homogeneous data, as highlighted in works such as [99, 52]. More recent research has expanded the scope of federated learning to encompass heterogeneous data settings (non-iid) and non-convex models, such as deep neural networks. In instances where datasets across various worker nodes exhibit homogeneity, FedAvg simplifies to local SGD, as outlined in [120]. Recent studies, as documented in [105, 51], explore FedAvg with partial worker node participation, employing $O(1)$ local update iterations and batch sizes. The resulting sample and communication complexities are both $O(\epsilon^{-4})$. In the works of [108, 109], Parallel Restarted SGD and Momentum SGD are introduced, demonstrating that both methods require $O(\varepsilon^{-4})$ samples and $O(\varepsilon^{-3})$ communication rounds to achieve an ε-stationary solution. The SCAFFOLD approach, proposed in [51], utilizes control variates to address 'client-drift'

in the presence of heterogeneous data, achieving identical sample and communication complexities as FedAvg. FedProx, introduced by [65], employs a penalty-based method to reduce communication complexity to $O(\varepsilon^{-2})$, with its analysis contingent on a gradient similarity assumption that constrains data heterogeneity by requiring all minima of $f(x)$ to also be minima of $f_i(x)$. Subsequently, FedPD, presented in [114], is proposed to relax this assumption.

Correct Client Drift is a big issue in federated learning. Inspired by optimization strategies like SVRG [49] and SAGA [23], inter-client variance reduction techniques [1, 51, 72] have been introduced in the context of Federated Learning (FL). These techniques aim to enhance FL by rectifying local training using anticipated local and global update directions. Notably, these methods are typically evaluated using convex or simpler non-convex models and objectives. Following this, several momentum-based federated learning (FL) algorithms have been introduced. For instance, [104] presents a momentum fusion technique to synchronize the server and local momentum buffers, albeit without a reduction in complexity. Leveraging variance reduction technology, Fed-GLOMO [22] requires a sample complexity and communication complexity both scaling as $O(\varepsilon^{-3})$. Notably, the sample complexity aligns with the optimal complexity of centralized non-convex stochastic optimization algorithms [27, 21]. More recently, STEM and FAFED, as proposed in [53, 101], employ a momentum-assisted stochastic gradient direction for both worker nodes and central server updates. This approach further diminishes communication rounds to $O(\varepsilon^{-2})$ while maintaining the same sample cost of $O(\varepsilon^{-3})$.

However, when it comes to the practical training of intricate ML models, the effectiveness of variance reduction techniques is called into question. Research such as [24] demonstrates that these techniques tend to underperform in the context of deep learning. This is primarily because the application of variance reduction to correct stochastic gradients is often rendered ineffective in deep learning due to prevalent augmentation strategies such as batch normalization [47] and dropout [94], among others.

2.5 Foundation Models

Foundation models, also known as large pre-trained neural network-based models. They have been trained on vast amounts of data to understand and generate data. These models are a type of transfer learning, where a model pre-trained on a large and diverse dataset is fine-tuned for specific downstream tasks. Foundation models serve as a starting point for a wide range of natural language processing machine learning tasks and can be further adapted or fine-tuned for more specific applications. For example, the CLIP [87] (Contrastive Language–Image Pretraining) model is a deep learning architecture developed by OpenAI that learns to associate images and their corresponding textual descriptions in a way that enables it to understand and generate cross-modal information. In other words, CLIP is designed to bridge the gap between images and text and has outstanding zero-shot ability.

Some existing works study the role of pre-trained Transformer models in Federated Learning (FL), [14, 78, 116], specifically in terms of effectively refining these pre-trained models within the FL framework and the potential advantages that FL users could gain from this innovative approach. However, there is a conflict between the foundation model and the edge device.

2.6 Model Distillation

Knowledge distillation serves as an instructional technique, transferring valuable insights and generalization capabilities from a trained teacher model to a student model [43, 3, 115, 113]. In the realm of Federated Learning (FL), [75] investigates adaptable aggregation methods with ensemble distillation at the server, while [90] employs an auxiliary dataset to weigh and ensemble local models from individual clients. FedDistill [91] extracts statistics related to the logit-vector from different client models and shares them among remaining clients for improved distillation. In a data-free knowledge distillation approach, [118] trains a generative model at the server, leveraging client information to create synthetic data for training client models. [20] introduces a co-distillation-based personalized

FL method, enabling cross-architecture training.

In our approach, we investigate the impact of knowledge distillation, as proposed by [43], on the performance of small-scale client models. Importantly, we achieve this without relying on excessive data, augmentations, or model sharing to uphold privacy. We aim to offer insights into the effective utilization of foundational models in the context of Federated Learning.

3.0 Task 1: Dynamic Model Screening Algorithms for Distributed Training

3.1 Background

In this chapter, we focus on the convex model compression with model screening. We also consider the distributed training and explore the method to reduce model dimension and speed up training. The work in this chapter was published in IJCAI 2022 [6].

Let local data $A = [a_1, \cdots, a_n]^\top \in \mathbb{R}^{n \times p}$, we consider the following composite optimization problem:

$$\min_{x \in \mathbb{R}^p} P(x) := \mathcal{F}(x) + \lambda \Omega(x), \tag{3-1}$$

where x is the model coefficient, $\Omega(x)$ is the block-separable sparsity-inducing norm, $\mathcal{F}(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(a_i^\top x)$ is the loss, and λ is the regularization parameter. We denote $\mathcal{F}_i(x) = f_i(a_i^\top x)$ for simplicity. Given partition $\mathcal{G}$ of the model coefficients, we denote the sub-matrix of A with the columns of $\mathcal{G}_j$ as $A_j \in \mathbb{R}^{n \times |\mathcal{G}_j|}$ and have $\Omega(x) = \sum_{j=1}^{q} \Omega_j(x_{\mathcal{G}_j})$.

In this part, we study sparsity regularized convex models with $l1$ norm in the distributed setting. We introduce the method known as distributed dynamic safe screening (DDSS), subsequently demonstrating its application to both shared-memory and distributed-memory architectures. Table 1 show the main different between our method and counterparts.

Table 1: Summary of dynamic safe screening methods. "Model" refers to the model it can solve where "MTL & MLR" means multi-task Lasso and multinomial logistic regression. "Safe" represents there are no active features eliminated. "Generalized" means whether it is limited to a specific model. "Distributed" represents whether it can work on distributed-memory architecture. "Scalablity" represents whether it is scalable with sample size n.

Reference	**Model**	**Safe**	**Generalized**	**Distributed**	**Scalablity**
[29]	Lasso	Yes	No	No	No
[84]	Sparse-Group Lasso	Yes	No	No	No
[93]	Sparse SVM	Yes	No	No	No
[83]	MTL & MLR	Yes	No	No	No
[88]	Proximal Weighted Lasso	Yes	No	No	No
[5]	OWL Regression	Yes	No	No	No
DDSS (Ours)	Problem (3-1)	Yes	Yes	Yes	Yes

3.2 Distributed Dynamic Safe Screening

3.2.1 Naive Implementation of DDSS

To harness the inherent sparsity in the model coefficients, we introduce an initial implementation of the distributed dynamic safe screening (DDSS) approach on a shared-memory architecture in Algorithm 1. Subsequently, we extend this implementation to a distributed-memory architecture in Algorithms 2 and Algorithm 3. Our explanatory emphasis primarily centers on the shared-memory architecture for illustrative purposes. During the training process, DDSS adeptly tackles the problem by systematically reducing its size through the removal of irrelevant features. It can compress the model size, reduce the commutation overhead, and speed up the model training.

Algorithm 1 Sha-DDSS-Naive

Input: $x^0_{\mathcal{B}_0} \in \mathbb{R}^p$, step size η, inner loops K
for $s = 0$ **to** $S - 1$ **do**
  All threads parallelly compute $\nabla \mathcal{F}(x^0_{\mathcal{B}_s})$
  Compute y^s by Equation (3-2) and $\mathcal{B}_{s+1}$ from $\mathcal{B}_s$ by Equation (3-3)
  Update $A_{\mathcal{B}_{s+1}}, x^0_{\mathcal{B}_{s+1}}$
  For each thread, do:
  for $t = 0$ **to** $K - 1$ **do**
    Read $\hat{x}^t_{\mathcal{B}_{s+1}}$ from the shared memory
    Randomly sample i from $\{1, 2, \ldots, n\}$
    $v^s_t = \nabla f_i(a^\top_{i,\mathcal{B}_{s+1}} \hat{x}^t_{\mathcal{B}_{s+1}})$
    Update $x^{t+1}_{\mathcal{B}_{s+1}} = \mathrm{prox}_{\eta\lambda\Omega}(\hat{x}^t_{\mathcal{B}_{s+1}} - \eta v^s_t)$
  end for
  $x_{\mathcal{B}_{s+1}} = x^K_{\mathcal{B}_{s+1}}, x^0_{\mathcal{B}_{s+1}} = x_{\mathcal{B}_{s+1}}$
end for

To be precise, Algorithm 1 consists of two loops. We represent the original problem as P_0, and the full feature set is denoted as $\mathcal{B}_0$. During the s-th iteration of the outer loop, we denote the active feature set as $\mathcal{B}_s$ and assume that $\mathcal{B}_s$ comprises q_s feature active blocks with a total of p_s active features. Consequently, the sub-problem P_s is defined over the set $\mathcal{B}_s$. The dual y^s can then be computed as follows:

$$y^s = -\nabla \mathcal{F}(x^0_{\mathcal{B}_s}) / \max(1, \Omega^D(A^\top_{\mathcal{B}_s} \nabla \mathcal{F}(x^0_{\mathcal{B}_s}))/\lambda), \tag{3-2}$$

where dual norm $\Omega^D(u) = \max_{\Omega(v)\leq 1} \langle v, u \rangle$.

With the obtained dual variable, we can eliminate inactive feature blocks (see details in [83, 5]) for $\forall j \in \mathcal{B}_s$ as

$$\Omega^D_j(A^\top_j y^s) + \Omega^D_j(A_j)\sqrt{2L(P(x^0_{\mathcal{B}_s}) - D(y^s))} < n\lambda \tag{3-3}$$

to update $\mathcal{B}_{s+1}$ where L is the Lipschitz constant. The dual objective $D(y^s)$ of P_s can be computed as:

$$\begin{aligned} &\max_{y^s} D(y^s) := -\frac{1}{n}\sum\nolimits_{i=1}^{n} f_i^*(-y_i^s) \\ s.t. \quad &\Omega_j^D(A_j^\top y^s) \leq n\lambda, \quad \forall j \in 1, \ldots, q_s \end{aligned} \tag{3-4}$$

Within the inner loop, all updates are executed on the constant feature set $\mathcal{B}_{s+1}$. Initially, each thread independently retrieves $\hat{x}^t_{\mathcal{B}s+1}$ from the shared memory and randomly selects a sample i to calculate the stochastic gradient over $\mathcal{B}_{s+1}$. Subsequently, the proximal step is performed based on this stochastic gradient. By leveraging equation (3-3), we ensure a continuous reduction in both the model and parameter sizes. This results in an accelerated training process achieved through the effective exploitation of model sparsity and reduced communication cost. Since any variable x_i eliminated via the screening process must be zero in the optimal solution, this approach ensures the safety of the training procedure.

Algorithm 2 Dis-DDSS-Naive (Server Node)

for $s = 0$ **to** $S - 1$ **do**
  flag = True
  Broadcast flag and $x^0_{\mathcal{B}_s}$ to all workers
  Receive gradients from all workers
  $\nabla\mathcal{F}(x^0_{\mathcal{B}_s}) = \frac{1}{n}\sum_{k=1}^{l} \nabla\mathcal{F}^k(x^0_{\mathcal{B}_s})$
  Compute y^s by (3-2)
  Update $\mathcal{B}_{s+1} \subseteq \mathcal{B}_s$ by (3-3)
  Broadcast $\mathcal{B}_{s+1}$ and $\nabla\mathcal{F}(x^0_{\mathcal{B}_s})$ to all workers
  flag = False
  Broadcast flag to all workers
  for $t = 0$ **to** $K - 1$ **do**
    Receive v_t^s from worker
    $x^{t+1}_{\mathcal{B}_{s+1}} = \text{prox}_{\eta\lambda\Omega}(x^t_{\mathcal{B}_{s+1}} - \eta v_t)$
  end for
  $x_{\mathcal{B}_{s+1}} = x^K_{\mathcal{B}_{s+1}}, x^0_{\mathcal{B}_{s+1}} = x_{\mathcal{B}_{s+1}}$
end for

Algorithm 3 Dis-DDSS-Naive (Worker Node k)

if flag = True **then**
  Receive $x^0_{\mathcal{B}_s}$ from server
  Compute and send full gradient $\nabla \mathcal{F}^k(x^0_{\mathcal{B}_s}) = \sum_{i \in n_k} \nabla f_i(a^\top_{i,\mathcal{B}_s} x^0_{\mathcal{B}_s})$
  Receive $\mathcal{B}_{s+1}$ from server
  Update $A_{i \in n_k, \mathcal{B}_{s+1}}, x^0_{\mathcal{B}_{s+1}}$
else
  Receive $x^{d(t)}_{\mathcal{B}_{s+1}}$ from server
  Randomly sample i from $\{1, 2, \ldots, n_k\}$
  Compute $v^s_t = \nabla f_i(a^\top_{i,\mathcal{B}_{s+1}} x^{d(t)}_{\mathcal{B}_{s+1}})$
  Send v^s_t to server
end if

3.2.2 Variance Reduction on the Active Set

Nonetheless, the variance of gradient estimation introduced by the stochastic sampling in Algorithm 1 fails to converge towards zero. Consequently, the necessity arises to employ a diminishing step size, yielding only marginal progress with each update. As a result, even in scenarios where P exhibits strong convexity, Algorithm 1 is limited to achieving a sublinear rate of convergence.

Given that the outer loop has already computed the full gradient for the elimination step, drawing inspiration from the variance-reduced technique in [103, 61], we can refine the gradient estimation over $\mathcal{B}_{s+1}$ by incorporating the exact gradient from the outer loop. This adjustment can be made without incurring additional computational costs, and is expressed as:

$$v^s_t \quad = \quad \nabla f_i(a^\top_{i,\mathcal{B}_{s+1}} \hat{x}^t_{\mathcal{B}_{s+1}}) - \nabla f_i(a^\top_{i,\mathcal{B}_{s+1}} x^0_{\mathcal{B}_{s+1}}) + \nabla \mathcal{F}(x^0_{\mathcal{B}_{s+1}}), \tag{3-5}$$

which can ensures the asymptotic convergence of the stochastic gradient variance towards zero. Consequently, it becomes feasible to employ a constant step size, thereby enabling substantial progress with each iteration and ultimately attaining a linear convergence rate for strongly convex functions.

3.2.3 Sparse Proximal Gradient Update

In real-world scenarios, sparsity (the value of a specific feature is 0) is prevalent in large-scale datasets. To capitalize on the sparsity of the dataset, it suffices to update only the blocks that contain nonzero partial gradients. Consequently, certain blocks may be updated more frequently than others. Drawing inspiration from [58] for Proximal SAGA, we introduce a block-wise reweighting matrix to perform a weighted projection on the blocks. Specifically, we define Ψ_i as the set of blocks that intersect with the nonzero coefficients of ∇f_i. Let $n_{\mathcal{G}}$ denote the number of occurrences of $\mathcal{G} \in \Psi_i$, and if $n_{\mathcal{G}} > 0$, we define $d_{\mathcal{G}} = n/n_{\mathcal{G}}$. Conversely, we can disregard that block directly. Consequently, we can define a diagonal matrix $[D_i]_{\mathcal{G},\mathcal{G}} = d_{\mathcal{G}} I_{|\mathcal{G}|}$ for each block i. Thus, the gradient over the set $\mathcal{B}_{s+1}$ can be expressed as:

$$v_t^s = \nabla f_i(a_{i,\mathcal{B}_{s+1}}^\top \hat{x}_{\mathcal{B}_{s+1}}^t) - \nabla f_i(a_{i,\mathcal{B}_{s+1}}^\top x_{\mathcal{B}_{s+1}}^0) + D_{i,\mathcal{B}_{s+1}} \nabla \mathcal{F}(x_{\mathcal{B}_{s+1}}^0). \tag{3-6}$$

Hence, it is sufficient to compute a sparse gradient and perform a sparse update solely over the active set, leading to a further reduction in computational costs.

At the same time, the proximal operator of the original Problem (3-1) is computed as

$$\text{prox}_{\eta\lambda\Omega}(x') = \arg\min_{x \in \mathbb{R}^p} \frac{1}{2\eta} \|x - x'\|^2 + \lambda\Omega(x). \tag{3-7}$$

In contrast, this method typically requires updating all coordinates for each iteration. Taking into account the sparsity of the dataset, it becomes apparent that updating only the blocks containing nonzero partial gradients is sufficient. Consequently, utilizing the reweighting matrix D, we employ a block-wise weighted norm $\phi_i(x) = \sum_{\mathcal{G} \in \Psi_i} d_{\mathcal{G}} \Omega_{\mathcal{G}}(x)$ to replace $\Omega(x)$.

It is worth noting that it is straightforward to verify that $\mathbb{E}[\phi_i(x)] = \Omega(x)$. Therefore, the computation of the new sparse proximal operator is as follows:

$$\text{prox}_{\eta\lambda\phi_i}(x') = \arg\min_{x \in \mathbb{R}^p} \frac{1}{2\eta} \|x - x'\|^2 + \lambda\phi_i(x). \tag{3-8}$$

Given that updates are exclusively applied to the blocks within Ψ_i, a subset that can be significantly smaller than the one necessitated for a complete pass over p coordinates due to sparsity, substantial savings in computational and memory expenses can be realized. In essence, the implementation leverages sparse gradient updates and sparse proximal operators to expedite training, capitalizing on the inherent sparsity within the dataset.

3.2.4 Decoupled Proximal Update

In the context of distributed-memory architecture, multiple workers calculate gradients and transmit them to a central server. Subsequently, the server performs the proximal operator computation. However, in instances where the proximal step demands considerable time, this process executed by the server can emerge as a computational bottleneck for the entire algorithm. Addressing this, [69] introduced a decoupled technique that redistributes the computational burden of the proximal step to the workers. This approach ensures that the server engages in relatively simpler addition computations, leading to a sublinear convergence rate and outperforming the coupled method.

In our algorithm, to alleviate the computational load on the server, the proximal mapping step is carried out by the workers, leaving the server to execute element-wise computations. The workers perform the proximal operator as follows:

$$x^{t+1}_{\mathcal{B}_{s+1}} = \text{prox}_{\eta\lambda\phi_i}(x^t_{\mathcal{B}_{s+1}} - \eta v^s_t), \tag{3-9}$$

and send the difference

$$\delta^s_t = \text{prox}_{\eta\lambda\phi_i}(x^t_{\mathcal{B}_{s+1}} - \eta v^s_t) - x^t_{\mathcal{B}_{s+1}}, \tag{3-10}$$

involving the parameter $x^t_{\mathcal{B}_{s+1}}$ and the output of the proximal operator to the server. Consequently, the server only performs straightforward addition computations, rendering the algorithm well-suited for parallelization to attain a linear speedup property. Moreover, acceleration can be achieved by increasing the number of workers.

3.3 DDSS on Shared-Memory Architecture

Algorithm 4 Sha-DDSS

for $s = 0$ **to** $S - 1$ **do**
  All threads parallelly compute $\nabla \mathcal{F}(x^0_{\mathcal{B}_s})$
  Compute y^s by (3-2) and $\mathcal{B}_{s+1}$ from $\mathcal{B}_s$ by (3-3)
  Update $A_{\mathcal{B}_{s+1}}, x^0_{\mathcal{B}_{s+1}}, \nabla(x^0_{\mathcal{B}_{s+1}})$
  For each thread, do:
  for $t = 0$ **to** $K - 1$ **do**
    Read $\hat{x}^t_{\mathcal{B}_{s+1}}$ from the shared memory
    Randomly sample i from $\{1, 2, \ldots, n\}$
    Compute v^s_t by (3-6)
    $\delta^s_t = \text{prox}_{\eta\lambda\phi_i}(\hat{x}^t_{\mathcal{B}_{s+1}} - \eta v^s_t) - \hat{x}^t_{\mathcal{B}_{s+1}}$
    $x^{t+1}_{\mathcal{B}_{s+1}} = x^t_{\mathcal{B}_{s+1}} + \delta^s_t$
  end for
  $x_{\mathcal{B}_{s+1}} = x^K_{\mathcal{B}_{s+1}}, x^0_{\mathcal{B}_{s+1}} = x_{\mathcal{B}_{s+1}}$
end for

In a shared-memory architecture, our Sha-DDSS algorithm is presented in Algorithm 4. Assuming we have l cores, during the outer loop, all threads parallelly calculate $\nabla \mathcal{F}(x^0_{\mathcal{B}s})$ and y^s, and perform the elimination. Following the formation of the new set $\mathcal{B}_{s+1}$, we update $A_{\mathcal{B}_{s+1}}$, $x^0_{\mathcal{B}_{s+1}}$, and $\nabla \mathcal{F}(x^0_{\mathcal{B}_{s+1}})$. Within the inner loop, the algorithm optimizes over $\mathcal{B}_{s+1}$, with multiple threads asynchronously updating the parameters. This asynchronous operation allows parameters to be read and written without the need for locks. Specifically, each thread inconsistently reads $\hat{x}^t_{\mathcal{B}_{s+1}}$ from shared memory and then selects a sample i from $1, 2, \ldots, n$. As expressed in Equation (3-6), we compute the gradient $v^s t$ over $\mathcal{B}_{s+1}$. Subsequently, we perform the proximal step, compute the update δ^s_t, and incorporate it into the shared memory.

Significantly, in the s-th iteration, our Algorithm 4 efficiently addresses sub-problem P_{s+1} within the sparse model structure, yielding a notable advantage over training the entire model. This translates to the computation of full gradients at the s-th iteration involving only

p_s coefficients, a considerably smaller number compared to the practical complexity of $O(p)$. Additionally, leveraging the sparsity inherent in the dataset enables us to execute sparse gradient updates and sparse proximal updates, particularly advantageous for handling large-scale real-world datasets. Moreover, the reduction of gradient variance allows us to employ a constant step size, enhancing convergence and ultimately achieving a linear convergence rate for strongly convex functions. Finally, the asynchronous operation of all threads proves to be highly efficient and easily parallelizable, effectively minimizing the computation and memory costs associated with large-scale training.

3.4 DDSS on Distributed-Memory Architecture

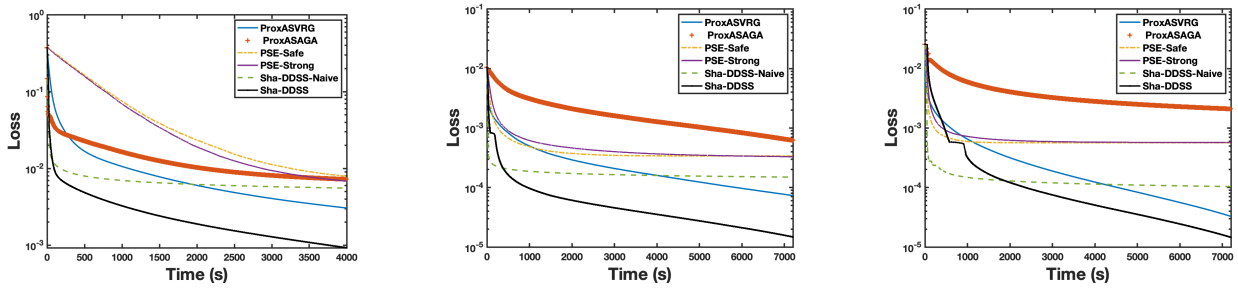
In a distributed-memory architecture, our Dis-DDSS algorithm is succinctly outlined in Algorithm 5 and Algorithm 6. In Dis-DDSS, assuming the presence of one server node and l local worker nodes, each worker storing n_k samples, the following procedures unfold. When the flag is set to True, within the outer loop of the server node, the server broadcasts both the flag and $x^0_{\mathcal{B}_s}$ to the workers. Subsequently, at each worker node, worker k receives $x^0_{\mathcal{B}_s}$ from the server, calculates the gradient over n_k samples, and transmits the result back to the server node. Once all gradients from the workers are received, the server node computes the full gradients and dispatches them to the workers. Utilizing Equation (3-2), the server node computes y^s and executes the elimination process to derive $\mathcal{B}_{s+1}$ before transmitting it to the workers. Each worker node, upon receiving $\nabla\mathcal{F}(x^0_{\mathcal{B}_s})$ and $\mathcal{B}s+1$ from the server, updates $A_{i\in n_k,\mathcal{B}_{s+1}}$, $x^0_{\mathcal{B}_{s+1}}$, and $\nabla\mathcal{F}(x^0_{\mathcal{B}_{s+1}})$ accordingly.

Algorithm 5 Dis-DDSS (Server Node)

for $s = 0$ **to** $S - 1$ **do**
  flag = True
  Broadcast flag and $x^0_{\mathcal{B}_s}$ to all workers
  Receive gradients from all workers
  $\nabla\mathcal{F}(x^0_{\mathcal{B}_s}) = \frac{1}{n}\sum_{k=1}^{l}\nabla\mathcal{F}^k(x^0_{\mathcal{B}_s})$
  Compute y^s by (3-2)
  Update $\mathcal{B}_{s+1} \subseteq \mathcal{B}_s$ by (3-3)
  Broadcast $\mathcal{B}_{s+1}$ and $\nabla\mathcal{F}(x^0_{\mathcal{B}_s})$ to all workers
  flag = False
  Broadcast flag to all workers
  for $t = 0$ **to** $K - 1$ **do**
    Receive δ^s_t from one worker
    Update $x^{t+1}_{\mathcal{B}_{s+1}} = x^t_{\mathcal{B}_{s+1}} + \delta^s_t$
  end for
  $x_{\mathcal{B}_{s+1}} = x^K_{\mathcal{B}_{s+1}}, x^0_{\mathcal{B}_{s+1}} = x_{\mathcal{B}_{s+1}}$
end for

Algorithm 6 Dis-DDSS (Worker Node k)

if flag = True **then**
  Receive $x^0_{\mathcal{B}_s}$ from server
  Compute and send gradient $\nabla\mathcal{F}^k(x^0_{\mathcal{B}_s}) = \sum_{i\in n_k}\nabla f_i(a^\top_{i,\mathcal{B}_s}x^0_{\mathcal{B}_s})$
  Receive $\nabla\mathcal{F}(x^0_{\mathcal{B}_s})$ from server
  Receive $\mathcal{B}_{s+1}$ from server
  Update $A_{i\in n_k,\mathcal{B}_{s+1}}, x^0_{\mathcal{B}_{s+1}}, \nabla\mathcal{F}(x^0_{\mathcal{B}_{s+1}})$
else
  Receive $x^{d(t)}_{\mathcal{B}_{s+1}}$ from server
  Randomly sample i from $\{1, 2, \ldots, n_k\}$
  Compute $v^s_t = \nabla f_i(a^\top_{i,\mathcal{B}_{s+1}}x^{d(t)}_{\mathcal{B}_{s+1}}) - \nabla f_i(a^\top_{i,\mathcal{B}_{s+1}}x^0_{\mathcal{B}_{s+1}}) + D_{i,\mathcal{B}_{s+1}}\nabla\mathcal{F}(x^0_{\mathcal{B}_{s+1}})$
  Send $\delta_t = \mathrm{prox}_{\eta\lambda\phi_i}(x^{d(t)}_{\mathcal{B}_{s+1}} - \eta v^s_t) - x^{d(t)}_{\mathcal{B}_{s+1}}$ to server
end if

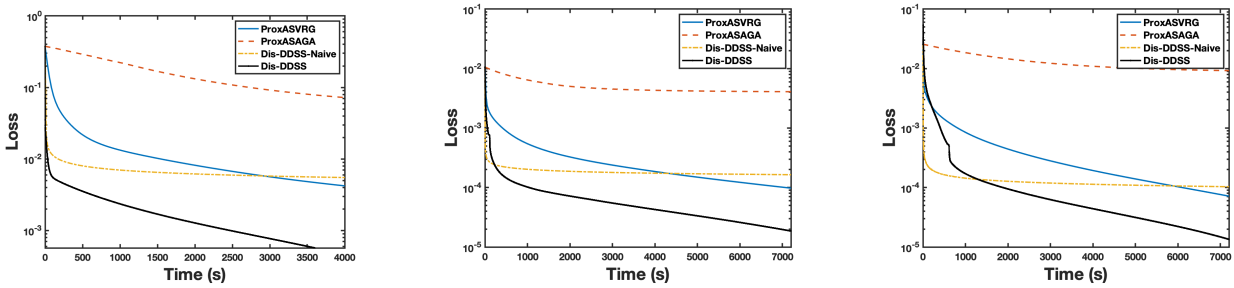


(a) KDD2010 (b) Avazu-app (c) Avazu-site

Figure 1: Convergence results on shared-memory architecture with 8 threads.

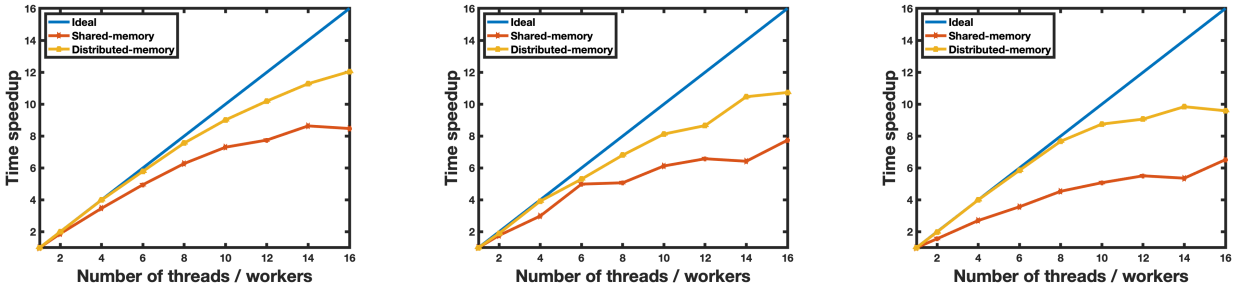
In the case where the flag is set to False, the server disseminates the flag to the workers. Subsequently, the algorithm optimizes over $\mathcal{B}_{s+1}$ at the worker nodes, where multiple workers independently update the parameters asynchronously. Specifically, each worker receives the outdated parameter $x_{\mathcal{B}_{s+1}}^{d(t)}$ from the server. The worker selects a sample i from $1, 2, \ldots, n_k$ and computes v_t^s over $\mathcal{B}_{s+1}$ using Equation (3-6). Employing the decoupling strategy, the worker calculates the proximal step and updates the parameter locally. Finally, the worker sends the update to the server. In the inner loop of the server node, leveraging the update information δ_t^s from workers, $x_{\mathcal{B}_{s+1}}^{t+1}$ is updated via a simple addition computation.

Firstly, akin to Algorithm 4, our Algorithm 5 and Algorithm 6 demonstrate significant computational efficiency by capitalizing on the sparsity of the model and dataset while mitigating gradient variance. Secondly, the resource-intensive proximal step typically handled by the server is distributed across all workers, proving to be highly efficient and easily parallelizable. Thirdly, through elimination and sparse updates, the communication cost between the server and workers is notably reduced compared to full updates. Overall, this approach effectively minimizes computation, memory, and communication costs during training.



(a) KDD2010 (b) Avazu-app (c) Avazu-site

Figure 2: Convergence results on distributed-memory architecture with 8 workers.



(a) KDD2010 (b) Avazu-app (c) Avazu-site

Figure 3: Linear speedup property on shared-memory and distributed-memory architecture.

Table 2: Real-world datasets in the experiments.

Dataset	Sample Size	Attributes	Sparsity
KDD 2010	19,264,097	1,163,024	7×10^{-6}
Avazu-app	14,596,137	1,000,000	10^{-5}
Avazu-site	25,832,830	1,000,000	10^{-5}

3.5 Experiments

3.5.1 Setup

We perform a comprehensive comparative analysis of our approach against other competitive methods across three extensive datasets. Although DDSS showcases versatility across diverse scenarios, our specific emphasis is on Lasso for sparse regression. This targeted focus is in line with the widely acknowledged application of Lasso for feature screening. To elucidate, Lasso tackles the optimization problem as follows:

$$\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y_i - a_i^\top x)^2 + \lambda \|x\|_1. \tag{3-11}$$

On shared-memory architecture, we choose following asynchronous methods as baseline algorithms: 1) PSE-Strong: parallel strong screening in [62]; 2) PSE-Safe: parallel static safe screening in [62]; 3) ProxASAGA [85]; 4) ProxASVRG [81]; 5) Sha-DDSS-Naive; 6) Our Sha-DDSS. PSE-Safe and PSE-Strong are parallel static screening. ProxASAGA and ProxASVRG are popular asynchronous methods with linear convergence.

On distributed-memory architecture, we compare four asynchronous methods: 1) ProxASAGA [59]; 2) ProxASVRG [81]; 3) Dis-DDSS-Naive; 4) Our Dis-DDSS.

Three large-scale real-world benchmark datasets are used (described in Table 2). All the datasets are from LIBSVM [13], which can be found at [1].

[1] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

We implement all the compared methods using C++. OpenMP and OpenMPI serve as the parallel framework for shared-memory and distributed-memory architecture, respectively. The methods are executed on machines with 2.10 GHz Intel(R) Xeon(R) CPUs. In the implementation process, we choose the inner loop size, ranging from 2×10^3 to 2×10^6, and the step size, ranging from 10^{-11} to 10^{-13}, for each method to achieve optimal performance. The parameter λ is set as $4 \times 10^{-6}\lambda_{\max}$, $2 \times 10^{-3}\lambda_{\max}$, and $1 \times 10^{-3}\lambda_{\max}$ for the KDD 2010, Avazu-app, and Avazu-site datasets, respectively, where $\lambda_{\max}$ is a parameter indicating that, for all $\lambda \geq \lambda_{\max}$, x^* must be 0.

3.5.2 Experimental Results and Discussions

Figure 1 (a)-(c) illustrates the convergence results of different methods on shared-memory architecture with 8 threads for three datasets, respectively. Our Sha-DDSS-Naive method demonstrates rapid convergence in the initial stages, attributed to its screening ability, but experiences a slowdown later due to its sublinear convergence rate. The outcomes affirm that our Sha-DDSS method consistently achieves much faster convergence than other methods on shared-memory architecture. Similarly, Figure 2 (a)-(c) presents the convergence results of different methods on distributed-memory architecture with 8 workers for three datasets, respectively. These results also validate that our Dis-DDSS consistently achieves faster convergence than other methods on distributed-memory architecture.

This superior performance is attributed to our method's ability on both shared-memory and distributed-memory architecture to eliminate features by exploiting the model's sparsity, conduct efficient sparse updates by leveraging the dataset's sparsity, and attain a linear convergence rate by mitigating gradient variance. Additionally, our Dis-DDSS implements a decoupled proximal update to alleviate the server's workload and reduce communication costs. Finally, 8 shows the linear speedup property on shared-memory and distributed-memory architecture.

3.6 Theoretical Analysis

In this section, we present a theoretical analysis of the convergence and screening ability of DDSS specifically focused on shared-memory architecture. It is worth noting that this analysis can be readily extended to distributed-memory architecture. All details about proof is shown in this section.

3.6.1 Assumptions, Definitions, and Properties

Assumption 1 (Strong Convexity)**.** $\Omega(x)$ *is convex and block separable.* $\mathcal{F}(x)$ *is* μ*-strongly convex, i.e.,* $\forall x, x' \in \mathbb{R}^p$*, we have*

$$\mathcal{F}(x') \geq \mathcal{F}(x) + \nabla\mathcal{F}(x)^\top(x' - x) + \frac{\mu}{2}\|x' - x\|^2. \tag{3-12}$$

Assumption 2 (Lipschitz Smooth)**.** *Each* $\mathcal{F}_i(x)$ *is differentiable and Lipschitz gradient continuous with* L*, i.e.,* $\exists L > 0$*, such that* $\forall x, x' \in \mathbb{R}^p$*, we have*

$$\|\nabla\mathcal{F}_i(x) - \nabla\mathcal{F}_i(x')\| \leq L\|x - x'\|. \tag{3-13}$$

Assumption 3 (Bounded Overlapping)**.** *A bound* τ *exists on the number of overlapping iterations. This implies that each write operation at iteration* t *is assured to be completed in the memory before iteration* $t + \tau + 1$*.*

Remark 1. *Assumption 1 implies* $P(x)$ *is also* μ*-strongly convex. Assumption 2 implies that* $\mathcal{F}(x)$ *is also Lipschitz gradient continuous. We denote* $\kappa := L/\mu$ *as the condition number. Assumption 3 means the delay that asynchrony may cause is upper bounded. All the assumptions are commonly seen in asynchronous methods [85].*

Definition 1 (Block Sparsity)**.** *We represent the maximum frequency of occurrences, denoted as* Δ*, indicating the instances where a feature block belongs to the extended support. This can be formally defined as:* $\Delta = \max_{\mathcal{G}\in\mathcal{B}} |\{i : \Psi_i \ni \mathcal{G}\}|/n$*. We can verify that* $1/n \leq \Delta \leq 1$*.*

Property 1 (Independence). *We use the "after read" labeling in [58], which means we update the iterate counter after each thread fully reads the parameters. This means that* $\hat{x}^t_{\mathcal{B}_{s+1}}$ *is the* $(t+1)$*-th fully completed read. Given the "after read" global time counter, sample* i_r *is independent of* $\hat{x}^t_{\mathcal{B}_{s+1}}, \forall r \geq t$.

Property 2 (Unbiased Gradient Estimation). *Gradient* v_t *is an unbiased estimation of the gradient over set* $\mathcal{B}_{s+1}$ *at* $\hat{x}^t_{\mathcal{B}_{s+1}}$*, which is directly derived from Property 1.*

Property 3 (Atomic Operation). *The update* $x^{t+1}_{\mathcal{B}_{s+1}} = x^t_{\mathcal{B}_{s+1}} + \delta^s_t$ *to shared-memory in Algorithm 4 is coordinate-wise atomic, which can address the overwriting problem caused by other threads.*

3.6.2 Theoretical Results

We provide a related theoretical analysis in this part. We will show theorem results and remarks. Details about proofs are provided in the 3.6.3.

Theorem 1 (Convergence). *Suppose* $\tau \leq \frac{1}{10\sqrt{\Delta}}$*, step size* $\eta = \min\{\frac{1}{24\kappa L}, \frac{\kappa}{2L}, \frac{\kappa}{10\tau L}\}$*, inner loop size* $K = \frac{4\log 3}{\eta\mu}$*, then we have*

$$\mathbb{E}\left\|x_{\mathcal{B}_S} - x^*_{\mathcal{B}_S}\right\|^2 \leq (2/3)^S \left\|x_0 - x^*\right\|^2. \tag{3-14}$$

Remark 2. *Theorem 1 proves that DDSS achieves a linear convergence rate of* $\log(1/\epsilon)$.

Remark 3. *If* $\mathcal{F}(x)$ *is nonstrongly convex, We can make a slight adjustment to* $\Omega(x)$ *by introducing a small perturbation, e.g.,* $\mu_f\|x\|^2$ *for smoothing where* μ_f *is a positive parameter.* $\mathcal{B}(x) + \mu_f\|x\|^2$ *is regarded as the loss and then the loss is* μ_f*-strongly convex. Denote* $\kappa := L/\mu_f$*, suppose* $\tau \leq \frac{1}{10\sqrt{\Delta}}$*, let* $\eta = \min\{\frac{1}{24\kappa L}, \frac{\kappa}{2L}, \frac{\kappa}{10\tau L}\}$*,* $K = \frac{4\log 3}{\eta\mu_f}$*, we have* $\mathbb{E}\left\|x_{\mathcal{B}_S} - x^*_{\mathcal{B}_S}\right\|^2 \leq (2/3)^S \left\|x_0 - x^*\right\|^2$*. Similarly, the overall computational cost is also very efficient.*

Theorem 2 (Screening Ability). *Equicorrelation set is defined as* $\mathcal{B}^* := \{j \in \{1, 2, \ldots, q\} : \Omega^D_j(A_j^\top y^*) = n\lambda\}$*. Then, as DDSS converges, there exists an iteration number* $S_0 \in \mathbb{N}$*, s.t.* $\forall s \geq S_0$*, any variable block* $j \notin \mathcal{B}^*$ *is removed by DDSS almost surely.*

Remark 4. *Assuming the size of set $\mathcal{B}_s$ is p_s and p^* represents the size of the active features in $\mathcal{B}^*$, Theorem 2 indicates that p_s is decreasing, and as s approaches infinity, $\lim_{s\to+\infty} p_s = p^*$.*

Remark 5. *In summary, through elimination, the cost per iteration decreases from $O(p)$ to $O(p_s)$. Additionally, sparse updates specifically target the non-zero coefficients in set $\mathcal{B}_s$, further reducing the cost per iteration from $O(p_s)$ to $O(p'_s)$, where p'_s represents the size of non-zero coefficients. In scenarios with high dimensionality, we observe that $p^* \ll p$, $p_s \ll p$, and $p'_s \ll p$. Consequently, as p_s continuously decreases along with sparse updates, our DDSS effectively lowers the complexity from $O(p)$ to $O(r)$, where r denotes the mean of p'_s for $s = 1, 2, \ldots$. This substantial reduction in complexity significantly accelerates training in practical applications.*

3.6.3 Basic Lemmas

We provide details about the theoretical analysis of DDSS, including basic lemmas and the proof for all the theorems.

Lemma 1. *Suppose $\mathcal{F}$ is μ-strongly convex, we have:*

$$\langle \nabla\mathcal{F}(y) - \nabla\mathcal{F}(x), y - x\rangle \geq \frac{\mu}{2}\|y - x\|^2 + B_{\mathcal{F}}(x, y) \tag{3-15}$$

where $B_{\mathcal{F}}(x, y)$ is the Bregman divergence defined as $B_{\mathcal{F}}(x, y) := \mathcal{F}(x) - \mathcal{F}(y) - \langle \nabla\mathcal{F}(y), x - y\rangle$.

Proof. By strong convexity, for any x, y, we have:

$$\begin{aligned}
& \mathcal{F}(y) \geq \mathcal{F}(x) + \langle \nabla\mathcal{F}(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2 \\
\iff\quad & \langle \nabla\mathcal{F}(x), x - y\rangle \geq \frac{\mu}{2}\|y - x\|^2 + \mathcal{F}(x) - \mathcal{F}(y) \\
\iff\quad & \langle \nabla\mathcal{F}(x) - \nabla\mathcal{F}(y), x - y\rangle \geq \frac{\mu}{2}\|y - x\|^2 + \mathcal{F}(x) - \mathcal{F}(y) - \langle \nabla\mathcal{F}(y), x - y\rangle \\
\iff\quad & \langle \nabla\mathcal{F}(y) - \nabla\mathcal{F}(x), x - y\rangle \geq \frac{\mu}{2}\|x - y\|^2 + B_{\mathcal{F}}(x, y).
\end{aligned} \tag{3-16}$$

This finishes the proof. □

Lemma 2. *Suppose $\mathcal{F}_i$ is L-smooth and convex, we have:*

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla\mathcal{F}_i(x)-\nabla\mathcal{F}_i(y)\|^2\leq 2LB_{\mathcal{F}}(x,y). \tag{3-17}$$

Proof. Based on equation 5 of Lemma 4 in [117], we have

$$\|\nabla\mathcal{F}_i(x)-\nabla\mathcal{F}_i(y)\|^2\leq 2L\big(\mathcal{F}_i(x)-\mathcal{F}_i(y)-\langle\nabla\mathcal{F}_i(y),x-y\rangle\big). \tag{3-18}$$

Averaging with i, we have

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla\mathcal{F}_i(x)-\nabla\mathcal{F}_i(y)\|^2\leq 2L\big(\mathcal{F}(x)-\mathcal{F}(y)-\langle\nabla\mathcal{F}(y),x-y\rangle\big), \tag{3-19}$$

which is equivalent to

$$\langle\nabla\mathcal{F}(x)-\nabla\mathcal{F}(y),x-y\rangle\geq\frac{\mu}{2}\|x-y\|^2+B_{\mathcal{F}}(x,y). \tag{3-20}$$

This finishes the proof. □

Lemma 3. *Let x^* be the optimal solution of Problem (3-1), v_t^s is the sparse variance reduced gradient with sample i defined in (3-6), g is the the sparse gradient mapping for v_t^s and computed as $g=\frac{1}{\eta}(x-\text{prox}_{\eta\lambda\phi_i}(x-\eta v_t^s))$. Then, for any $\beta>0$ and $x\in\mathbb{R}^p$, we have:*

$$\langle g,x-x^*\rangle\geq-\frac{\eta}{2}(\beta-2)\|g\|^2-\frac{\eta}{2\beta}\left\|v_t^s-D_i\nabla\mathcal{F}\left(x^*\right)\right\|^2+\left\langle v_t^s-D_i\nabla\mathcal{F}\left(x^*\right),x-x^*\right\rangle. \tag{3-21}$$

Proof. Let $x^+=\text{prox}_{\eta\lambda\phi_i}(x-\eta v_t^s)$ and $x^*=\text{prox}_{\eta\lambda\phi_i}(x^*-\eta D\nabla\mathcal{F}(x^*))$, denote $\langle\cdot,\cdot\rangle_{(i)}$ as the inner product restricted to the blocks in Ψ_i and $\|\cdot\|_{(i)}$ as the norm restricted to the blocks in Ψ_i, we have:

$$\|x^+-x^*\|_{(i)}^2-\langle x^+-x^*,x-\eta v_t^s-x^*+\eta D\nabla\mathcal{F}(x^*)\rangle_{(i)}\leq 0, \tag{3-22}$$

which comes from the firm non-expansiveness of the proximal operator. Then we have:

$$\begin{aligned}
\langle \eta g, x - x^* \rangle &= \langle x - x^+, x - x^* \rangle_{(i)} \\
&= \|x - x^*\|_{(i)}^2 - \langle x^+ - x^*, x - x^* \rangle_{(i)} \\
&\geq \|x - x^*\|_{(i)}^2 - \langle x^+ - x^*, 2x - \eta v_t^s - 2x^* + \eta D \nabla \mathcal{F}(x^*) \rangle_{(i)} + \|x^+ - x^*\|_{(i)}^2 \\
&= \|x - x^+\|_{(i)}^2 + \langle x^+ - x^*, \eta v_t^s - \eta D \nabla \mathcal{F}(x^*) \rangle_{(i)} \\
&= \|x - x^+\|_{(i)}^2 + \langle x - x^*, \eta v_t^s - \eta D \nabla \mathcal{F}(x^*) \rangle_{(i)} - \langle x - x^+, \eta v_t^s - \eta D \nabla \mathcal{F}(x^*) \rangle_{(i)} \\
&\geq \Big(1 - \frac{\beta}{2}\Big) \|x - x^+\|_{(i)}^2 - \frac{\eta^2}{2\beta} \|v_t^s - D \nabla \mathcal{F}(x^*)\|_{(i)}^2 + \eta \langle v_t^s - D \nabla \mathcal{F}(x^*), x - x^* \rangle_{(i)} \\
&= \Big(1 - \frac{\beta}{2}\Big) \|x - x^+\|_{(i)}^2 - \frac{\eta^2}{2\beta} \|v_t^s - D_i \nabla \mathcal{F}(x^*)\|^2 + \eta \langle v_t^s - D_i \nabla \mathcal{F}(x^*), x - x^* \rangle \\
&= \Big(1 - \frac{\beta}{2}\Big) \|\eta g\|^2 - \frac{\eta^2}{2\beta} \|v_t^s - D_i \nabla \mathcal{F}(x^*)\|^2 + \eta \langle v_t^s - D_i \nabla \mathcal{F}(x^*), x - x^* \rangle \,, \quad (3\text{-}23)
\end{aligned}$$

where the first inequality is obtained by adding equation (3-22), the second inequality is obtained by Young's inequality $2\langle a, b \rangle \leq \frac{\|a\|^2}{\beta} + \beta \|b\|^2$ for arbitrary $\beta > 0$. Finally, the result finishes the proof. □

Lemma 4. *[58, Proposition 1] For any $u \neq t$, we have*

$$\mathbb{E}|\langle g_u, g_t \rangle| \leq \tfrac{\sqrt{\Delta}}{2} (\mathbb{E}\|g_u\|^2 + \mathbb{E}\|g_t\|^2) \,. \quad (3\text{-}24)$$

Lemma 5. *We have following estimations:*

$$\mathbb{E} \, \|\hat{x}_t^s - x_t^s\|^2 \leq \eta^2 (1 + \sqrt{\Delta}\tau) \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E} \, \|g_u^s\|^2 \quad (3\text{-}25)$$

$$\mathbb{E} \, \langle g_t^s, \hat{x}_t^s - x_t^s \rangle \leq \frac{\eta\sqrt{\Delta}}{2} \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E} \, \|g_u^s\|^2 + \frac{\eta\sqrt{\Delta}\tau}{2} \mathbb{E} \, \|g_t^s\|^2 \quad (3\text{-}26)$$

Proof. By Assumption 3, we have the following updates:

$$\hat{x}_t - x_t = \eta \sum_{u=(t-\tau)_+}^{t-1} G_u^t g(\hat{x}_u, \hat{\alpha}^u, i_u). \tag{3-27}$$

Thus, we have:

$$\begin{aligned}
\mathbb{E}\|\hat{x}_t - x_t\|^2 &\leq \eta^2 \sum_{u,v=(t-\tau)_+}^{t-1} \mathbb{E}|\langle G_u^t g_u, G_v^t g_v\rangle| \\
&\leq \eta^2 \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 + \eta^2 \sum_{\substack{u,v=(t-\tau)_+ \\ u\neq v}}^{t-1} \mathbb{E}|\langle G_u^t g_u, G_v^t g_v\rangle| \\
&\leq \eta^2 \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 + \eta^2 \sum_{\substack{u,v=(t-\tau)_+ \\ u\neq v}}^{t-1} \mathbb{E}|\langle g_u, g_v\rangle| \\
&\leq \eta^2 \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 + \eta^2\sqrt{\Delta}(\tau-1)_+ \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 \\
&= \eta^2\big(1+\sqrt{\Delta}(\tau-1)_+\big) \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 \\
&\leq \eta^2\big(1+\sqrt{\Delta}\tau\big) \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2,
\end{aligned} \tag{3-28}$$

where the fourth inequality is obtained by Lemma 4.

Taking the expectation of $\langle \hat{x}_t - x_t, g_t\rangle$, we have:

$$\begin{aligned}
\frac{1}{\eta}\mathbb{E}\langle \hat{x}_t - x_t, g_t\rangle &= \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\langle G_u^t g_u, g_t\rangle \\
&\leq \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}|\langle g_u, g_t\rangle| \\
&\leq \sum_{u=(t-\tau)_+}^{t-1} \frac{\sqrt{\Delta}}{2}(\mathbb{E}\|g_u\|^2 + \mathbb{E}\|g_t\|^2) \\
&\leq \frac{\sqrt{\Delta}}{2} \sum_{u=(t-\tau)_+}^{t-1} \mathbb{E}\|g_u\|^2 + \frac{\sqrt{\Delta}\tau}{2}\mathbb{E}\|g_t\|^2.
\end{aligned} \tag{3-29}$$

This finishes the proof. □

Lemma 6. *For any $x \in \mathbb{R}^p$ and gradient estimator v_t^s computed by sample i, we have*

$$\mathbb{E}\left\|v_t^s - D_i\nabla\mathcal{F}(x^*)\right\|^2 \leq 4L\mathbb{E}B_{\mathcal{F}}(\hat{x}_t^s, x^*) + 2L^2\mathbb{E}\|x_0^s - x^*\|^2 \tag{3-30}$$

Proof. Since v_t^s is Ψ_i-sparse, we have

$$\begin{aligned}
&\left\|v_t^s - D_i\nabla\mathcal{F}(x^*)\right\|^2 \\
=&\left\|v_t^s - D\nabla\mathcal{F}(x^*)\right\|_{(i)}^2 \\
=&\left\|\nabla\mathcal{F}_i(\hat{x}_t^s) - \nabla\mathcal{F}_i(x_0^s) + D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*)\right\|_{(i)}^2 \\
=&\left\|\nabla\mathcal{F}_i(\hat{x}_t^s) - \nabla\mathcal{F}_i(x^*) + D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*) - (\nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*))\right\|_{(i)}^2 \\
\leq&2\left\|\nabla\mathcal{F}_i(\hat{x}_t^s) - \nabla\mathcal{F}_i(x^*)\right\|_{(i)}^2 + 2\left\|D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*) - (\nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*))\right\|_{(i)}^2.
\end{aligned}$$

According to the support set of $\nabla\mathcal{F}_i$ and Lemma 2, we get

$$\mathbb{E}\left\|\nabla\mathcal{F}_i(\hat{x}_t^s) - \nabla\mathcal{F}_i(x^*)\right\|_{(i)}^2 \leq 2L\mathbb{E}B_{\mathcal{F}}(\hat{x}_t^s, x^*). \tag{3-31}$$

On the other hand,

$$\begin{aligned}
&\mathbb{E}\left\|D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*) - (\nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*))\right\|_{(i)}^2 \\
=\ &\mathbb{E}\left\|D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*)\right\|_{(i)}^2 + \mathbb{E}\left\|\nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*)\right\|_{(i)}^2 \\
&-2\mathbb{E}\langle D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*), \nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*)\rangle_{(i)} \\
=\ &\mathbb{E}\left\|\nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*)\right\|^2 + \mathbb{E}\left\langle D_i\nabla\mathcal{F}(x_0^s) - D_i\nabla\mathcal{F}(x^*), D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*)\right\rangle \\
&-2\mathbb{E}\langle D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*), \nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*)\rangle \\
=\ &\mathbb{E}\left\|\nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*)\right\|^2 + \mathbb{E}\left\langle \nabla\mathcal{F}(x_0^s) - \nabla\mathcal{F}(x^*), D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*)\right\rangle \\
&-2\mathbb{E}\langle D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*), \nabla\mathcal{F}(x_0^s) - \nabla\mathcal{F}(x^*)\rangle \\
\leq\ &\mathbb{E}\left\|\nabla\mathcal{F}_i(x_0^s) - \nabla\mathcal{F}_i(x^*)\right\|^2 \\
\leq\ &L^2\mathbb{E}\|x_0^s - x^*\|^2,
\end{aligned} \tag{3-32}$$

where the second equality is derived by the definition of D_i and the support of $\nabla\mathcal{F}_i$. In the third equality, we take expectation on i and use $\mathbb{E}D_i = I_p$. In the first inequality, we use the fact that D is a diagonal matrix with non-negative entries and hence $\langle\nabla\mathcal{F}(x_0^s) - \nabla\mathcal{F}(x^*), D\nabla\mathcal{F}(x_0^s) - D\nabla\mathcal{F}(x^*)\rangle \geq 0$. The last inequality comes from Assumption 2. Combining above inequalities, we complete the proof. □

3.6.4 Proof of Theorem 1

In this part, we will work on the proof of theorem 1 in the DDSS

Proof. At the $s-1$ iteration, we conduct the elimination step over set $\mathcal{B}_{s-1}$ in the outer loop. Since the eliminated variables are zeroes at the optimal, all the sub-problems have the same optimal solution. We have

$$\|x^*_{\mathcal{B}_s} - x^*_{\mathcal{B}_{s-1}}\|^2 = 0$$

where the norm is conducted on the corresponding coordinate and the eliminated variables in B_s are filled with 0. Meanwhile, denote $\tilde{\mathcal{B}_s} = \{j \in \mathcal{B}_{s-1} | j \notin \mathcal{B}_s\}$, we have

$$\|x^0_{\mathcal{B}_{s-1}} - x^*_{\mathcal{B}_{s-1}}\|^2 = \|x^0_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + \|x^0_{\tilde{\mathcal{B}_s}} - x^*_{\tilde{\mathcal{B}_s}}\|^2. \tag{3-33}$$

Considering $\|x^0_{\tilde{\mathcal{B}_s}} - x^*_{\tilde{\mathcal{B}_s}}\|^2 \geq 0$, we have

$$\|x^0_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 \leq \|x^0_{\mathcal{B}_{s-1}} - x^*_{\mathcal{B}_{s-1}}\|^2. \tag{3-34}$$

Moreover, according to the iteration in the inner loop, we have

$$\begin{aligned}
\|x^{t+1}_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 &= \|x^t_{\mathcal{B}_s} - \eta g^s_t - x^*_{\mathcal{B}_s}\|^2 \\
&= \|x^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + \|\eta g^s_t\|^2 - 2\eta\langle g^s_t, x^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\rangle \\
&= \|x^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + \|\eta g^s_t\|^2 - 2\eta\langle g^s_t, \hat{x}^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\rangle + 2\eta\langle g^s_t, \hat{x}^t_{\mathcal{B}_s} - x^t_{\mathcal{B}_s}\rangle
\end{aligned} \tag{3-35}$$

Applying Lemma 3 to $\mathcal{B}_s$, we obtain

$$\begin{aligned}
&\|x^{t+1}_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 \leq \|x^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + 2\eta\langle g^s_t, \hat{x}^t_{\mathcal{B}_s} - x^t_{\mathcal{B}_s}\rangle + \eta^2(\beta-1)\|g^s_t\|^2 \\
+ \quad &\tfrac{\eta^2}{\beta}\|v^{s-1}_t - D_{i,\mathcal{B}_s}\nabla\mathcal{F}(x^*_{\mathcal{B}_s})\|^2 - 2\eta\langle v^{s-1}_t - D_{i,\mathcal{B}_s}\nabla\mathcal{F}(x^*_{\mathcal{B}_s}), \hat{x}^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\rangle
\end{aligned} \tag{3-36}$$

Since $\mathbb{E}_i D_{i,\mathcal{B}_s} = I_{p_s}$, we have

$$\begin{aligned}
\mathbb{E}\langle v^{s-1}_t - D_{i,\mathcal{B}_s}\nabla\mathcal{F}(x^*_{\mathcal{B}_s}), \hat{x}^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\rangle &= \langle\nabla\mathcal{F}(\hat{x}^t_{\mathcal{B}_s}) - \nabla\mathcal{F}(x^*_{\mathcal{B}_s}), \hat{x}^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\rangle \\
&\geq \frac{\mu}{2}\|\hat{x}^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + B_{\mathcal{F}}(\hat{x}^t_{\mathcal{B}_s}, x^*_{\mathcal{B}_s}),
\end{aligned} \tag{3-37}$$

where the inequality is obtained by applying Lemma 1 to the sub-problem P_s. Combining above two inequalities we get

$$\begin{aligned}\|x^{t+1}_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 \leq & \quad \|x^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + 2\eta\langle g^s_t, \hat{x}^t_{\mathcal{B}_s} - x^t_{\mathcal{B}_s}\rangle + \eta^2(\beta-1)\|g^s_t\|^2 \\ + & \quad \tfrac{\eta^2}{\beta}\|v^{s-1}_t - D_{i,\mathcal{B}_s}\nabla\mathcal{F}(x^*_{\mathcal{B}_s})\|^2 - \eta\mu\|\hat{x}^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 - 2\eta B_{\mathcal{F}}(\hat{x}^t_{\mathcal{B}_s}, x^*_{\mathcal{B}_s}) \\ \leq & \quad (1-\tfrac{\eta\mu}{2})\|x^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + 2\eta\langle g^s_t, \hat{x}^t_{\mathcal{B}_s} - x^t_{\mathcal{B}_s}\rangle + \eta^2(\beta-1)\|g^s_t\|^2 \\ + & \quad \tfrac{\eta^2}{\beta}\|v^{s-1}_t - D_{i,\mathcal{B}_s}\nabla\mathcal{F}(x^*_{\mathcal{B}_s})\|^2 + \eta\mu\|\hat{x}^t_{\mathcal{B}_s} - x^t_{\mathcal{B}_s}\|^2 - 2\eta B_{\mathcal{F}}(\hat{x}^t_{\mathcal{B}_s}, x^*_{\mathcal{B}_s}) \quad (3\text{-}38)\end{aligned}$$

where in the second inequality we use $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$.

By considering Lemma 5 and Lemma 6 with $\mathcal{B}_s$, we have

$$\begin{aligned}&\mathbb{E}\|x^{t+1}_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 \\ \leq & (1-\frac{\eta\mu}{2})\mathbb{E}\|x^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + \frac{2L^2\eta^2}{\beta}\mathbb{E}\|x^0_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + \frac{4L\eta^2}{\beta}\mathbb{E}B_{\mathcal{F}}(\hat{x}^t_{\mathcal{B}_s}, x^*_{\mathcal{B}_s}) \\ & - 2\eta\mathbb{E}B_{\mathcal{F}}(\hat{x}^t_{\mathcal{B}_s}, x^*_{\mathcal{B}_s}) + \eta^2(\beta - 1 + \sqrt{\Delta}\tau)\mathbb{E}\left\|g^s_t\right\|^2 \\ & + (\eta^3\mu(1+\sqrt{\Delta}\tau) + \eta^2\sqrt{\Delta})\sum_{u=(t-\tau)_+}^{t-1}\mathbb{E}\left\|g^s_u\right\|^2. \quad (3\text{-}39)\end{aligned}$$

Because $4L\eta \leq 1$, $\sqrt{\Delta}\tau \leq \frac{1}{10}$ and $B_{\mathcal{F}}(\cdot,\cdot)$ is non-negative, let $\beta = \frac{1}{2}$, we have

$$\begin{aligned}\mathbb{E}\|x^{t+1}_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 \leq & \quad (1-\frac{\eta\mu}{2})\mathbb{E}\|x^t_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 + 4L^2\eta^2\mathbb{E}\|x^0_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 - \frac{2\eta^2}{5}\mathbb{E}\left\|g^s_t\right\|^2 \\ & + (2\eta^3\mu + \eta^2\sqrt{\Delta})\sum_{u=(t-\tau)_+}^{t-1}\mathbb{E}\left\|g^s_u\right\|^2. \quad (3\text{-}40)\end{aligned}$$

Applying recursion to above inequality and we can obtain

$$\mathbb{E}\|x^K_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 \leq \Big((1-\frac{\eta\mu}{2})^K + 8\kappa L\eta\Big)\mathbb{E}\|x^0_{\mathcal{B}_s} - x^*_{\mathcal{B}_s}\|^2 - \sum_{u=0}^{K-1} r^s_u\mathbb{E}\|g^s_u\|^2, \quad (3\text{-}41)$$

where

$$r^s_u = (1-\frac{\eta\mu}{2})^{K-1-u}\Big(\frac{2\eta^2}{5} - (2\eta^3\mu + \eta^2\sqrt{\Delta})\sum_{m=0}^{\tau-1}(1-\frac{\eta\mu}{2})^{-m}\Big). \quad (3\text{-}42)$$

Let $h(x) = \log(1+2ax) - x\log(1+a)$ for some positive a. We can see $h(0) = 0$ and $h'(x) = \frac{2a}{1+2ax} - \log(1+a)$. If $ax \leq \frac{1}{2}$, then we have $h'(x) \geq a - \log(1+a) \geq 0$. Therefore,

when $ax \leq \frac{1}{2}$, we always have $h(x) \geq 0$ and $(1+a)^x \leq 1+2ax$. Substitute a with $\frac{\eta\mu}{2-\eta\mu}$. Since $\eta\mu\tau \leq \frac{1}{10}$, we have $\frac{\eta\mu\tau}{2-\eta\mu} \leq \frac{1}{2}$.

Then we can estimate the lower bound of r_u^s:

$$r_u^s \geq (1-\frac{\eta\mu}{2})^{K-1-u}\Big(\frac{2\eta^2}{5} - (2\eta^3\mu + \eta^2\sqrt{\Delta})(\tau + \frac{\eta\mu\tau^2}{2-\eta\mu})\Big) \geq (1-\frac{\eta\mu}{2})^{K-1-u}\frac{7\eta^2}{100}, \quad (3\text{-}43)$$

where we also use $\eta\mu\tau \leq \frac{1}{10}$ and $\sqrt{\Delta}\tau \leq \frac{1}{10}$. As r_u^s is positive, we can drop the last term in equation (3-41).

As $\eta\mu \leq \frac{1}{2}$, it is easy to check that $\log(1+\frac{\eta\mu}{2-\eta\mu}) \geq \frac{\eta\mu}{2(2-\eta\mu)}$. When $K = \frac{4\log 3}{\eta\mu}$, we have $(1-\frac{\eta\mu}{2})^K \leq \frac{1}{3}$. Since $\eta \leq \frac{1}{24\kappa L}$, we can prove

$$\mathbb{E}\|x_{\mathcal{B}_s}^K - x_{\mathcal{B}_s}^*\|^2 \leq \frac{2}{3}\mathbb{E}\|x_{\mathcal{B}_s}^0 - x_{\mathcal{B}_s}^*\|^2. \quad (3\text{-}44)$$

Note $x_{\mathcal{B}_s}^K = x_{\mathcal{B}_s}$ and $x_{\mathcal{B}_{s-1}}^0 = x_{\mathcal{B}_{s-1}}$, combining (3-34), we have

$$\mathbb{E}\|x_{\mathcal{B}_s} - x_{\mathcal{B}_s}^*\|^2 \leq \frac{2}{3}\mathbb{E}\|x_{\mathcal{B}_{s-1}} - x_{\mathcal{B}_{s-1}}^*\|^2. \quad (3\text{-}45)$$

Apply recursion to the above inequality, note $x_{\mathcal{B}_0} = x_0$ and $x_{\mathcal{B}_0}^* = x^*$ we can obtain

$$\mathbb{E}\|x_{\mathcal{B}_S} - x_{\mathcal{B}_S}^*\|^2 \leq (\frac{2}{3})^S\mathbb{E}\|x_0 - x^*\|^2, \quad (3\text{-}46)$$

which finishes the proof. □

3.6.5 Proof of Theorem 2

In this part, we will work on the proof of theorem 2 in the DDSS.

Proof. Based on Theorem 1, we know our DDSS method has a converging sequence $\{x_{\mathcal{B}_s}\}$. As DDSS algorithm converges, because of the strong duality, the dual y^s and the intermediate duality gap $P(x_{\mathcal{B}_s}) - D(y^s)$ also converges. For any given ϵ, $\exists S_0$ such that $\forall s \geq S_0$, we have

$$\|y^s - y^*\|_2 \leq \epsilon, \tag{3-47}$$

and

$$\sqrt{2L(P(x_{\mathcal{B}_s}) - D(y^s))} \leq \epsilon. \tag{3-48}$$

almost surely.

For any $j \notin \mathcal{B}^*$, we have

$$\begin{aligned} & \Omega_j^D(A_j^\top y^s) + \Omega_j^D(A_j)\sqrt{2L(P(x_{\mathcal{B}_s}) - D(y^s))} \\ \leq\ & \Omega_j^D(A_j^\top(y^s - y^*)) + \Omega_j^D(A_j^\top y^*) + \Omega_j^D(A_j)\sqrt{2L(P(x_{\mathcal{B}_s}) - D(y^s))} \\ \leq\ & 2\Omega_j^D(A_j)\epsilon + \Omega_j^D(A_j^\top y^*) \end{aligned} \tag{3-49}$$

where the first inequality comes from the triangle inequality and the second inequality comes from (3-47) and (3-48).

Hence, if we choose

$$\epsilon < \frac{n\lambda - \Omega_j^D(A_j^\top y^*)}{2\Omega_j^D(A_j)}, \tag{3-50}$$

we can ensure the screening test $\Omega_j^D(A_j^\top y^s) + \Omega_j^D(A_j)\sqrt{2L(P(x_{\mathcal{B}_s}) - D(y^s))} < n\lambda$ holds for j, which means variable block j is eliminated at most at this iteration. In (3-50), since $j \notin \mathcal{B}^*$, it is easily to verify that $n\lambda - \Omega_j^D(A_j^\top y^*) > 0$. This finishes the proof. □

3.7 Summary

In this chapter, we propose the first distributed dynamic safe screening method for sparse models and apply it on shared-memory and distributed-memory architecture respectively. Theoretically, we prove that our proposed method can achieve a linear convergence rate with lower overall complexity. Moreover, we prove that our method can eliminate almost all the inactive variables in a finite number of iterations almost surely. Finally, extensive experimental results on benchmark datasets confirm the significant acceleration and linear speedup property of our method.

The main contributions of our work can be summarized as follows.

- We propose a new distributed dynamic safe screening framework for generalized sparse models, which is easy-to-implement on the both shared-memory and distributed-memory architecture. To the best of knowledge, this is the first work of distributed dynamic safe screening.
- We rigorously prove the proposed DDSS method can achieve linear convergence rate $O(\log(1/\epsilon))$, reduce the per-iteration cost from $O(p)$ to $O(r)$ where $r \ll p$, and finally achieve a lower overall computational complexity under the strongly convex condition.
- We prove almost sure finite time identification of the active set to confirm the effectiveness of our DDSS method. Finally, we empirically show that our proposed method can achieve significant acceleration and linear speedup properties.

4.0 Task 2: Auto-Train-Once: Controller Network Guided Automatic Network Pruning from Scratch

4.1 Background

In this chapter, we study the non-convex model compression as deep learning presents remarkable performance and all deep models are non-convex. The work in this chapter was published in CVPR2024 [100].

There are many directions to reduce the size of deep learning models. Structural pruning, as advocated in various studies such as [30], is a widely embraced strategy for reducing the size of deep neural networks (DNNs) due to its broad applicability and effectiveness. In contrast to weight pruning, structural pruning, especially channel pruning, is more conducive to hardware implementation as it eliminates the need for additional post-processing steps to achieve computational and storage efficiencies. Consequently, our focus is on employing structural pruning in DNNs. However, it is important to highlight that many existing pruning methods have notable limitations and involve a complex multi-stage process. The majority of current structural pruning techniques follow a three-stage protocol: (1) train a complete model from the ground up; (2) pinpoint redundant structures based on various criteria; (3) fine-tune or retrain the pruned model to recover performance. Various methods employ distinct criteria for the pruning process. Managing this multi-stage training process for DNNs demands considerable engineering efforts and specialized expertise. To streamline pruning methods, recent approaches such as OTO [16] (Only Train once) and its successor OTOv2 [17] propose an end-to-end training and pruning approach. These methods introduce the concept of zero-invariant groups (ZIGs) and concurrently train and prune models without relying on additional fine-tuning, simplifying the overall process.

Nevertheless, the straightforward training frameworks employed in OTO and OTOv2 present a challenge to model performance. These frameworks redefine the objective as a constrained regularization problem, and the local minima with superior generalization may be dispersed across diverse locations. While the augmented regularization in OTO penalizes

the mixed L1/L2 norm of all trainable parameters in zero-invariant groups (ZIGs), it confines the search space to converge around the origin point.

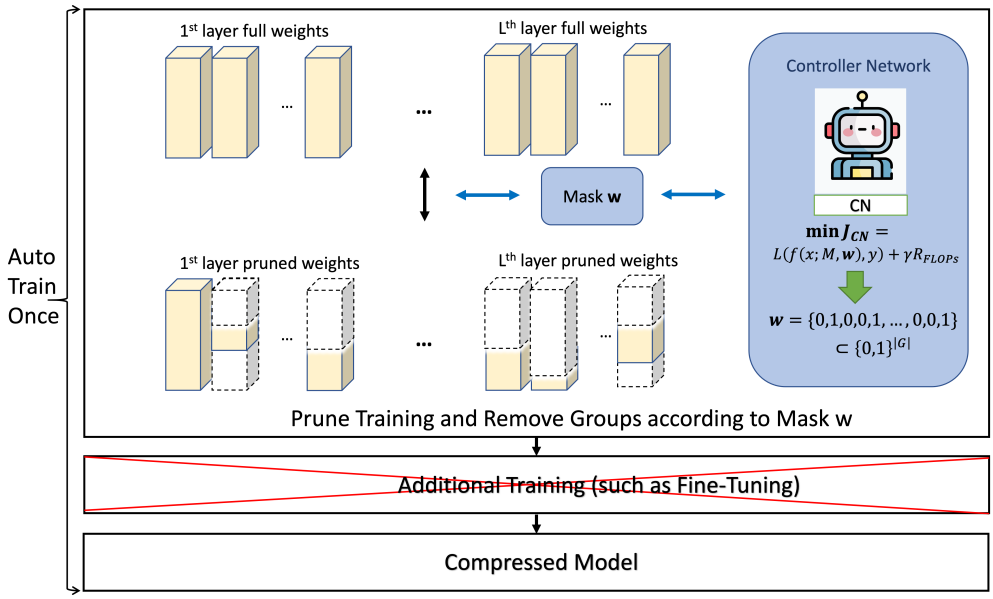


Figure 4: Overview of Auto-Train-Once (ATO). The controller network generates mask $\mathbf{w}$ based on the size of ZIGs $\mathcal{G}$ to guide the automatic network pruning of the target model and we remove variable groups according to mask $\mathbf{w}$ after training. Additional training (such as fine-tuning) is not required after model training and we can directly get the final compressed model.

OTOv2 addresses this limitation by enhancing OTO, constructing pruning groups in ZIGs based on salience scores to penalize only the trainable parameters in these groups. However, the dynamic nature of model variables during training and the statically selected pruning groups of the optimizer in the early training stages can lead to convergence issues with local optima and result in suboptimal final performance. These algorithmic design drawbacks impede a comprehensive convergence analysis. For example, OTO assumes the deep model to be a strongly convex function, and OTOv2 assumes a full gradient estimate at each iteration, which does not align with the practical settings of deep neural network (DNN) training.

In order to elevate model performance while preserving a comparable advantage, we

Table 3: 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	√	╳	-
Gradient projection	**General**	**(D)HSPG**	-

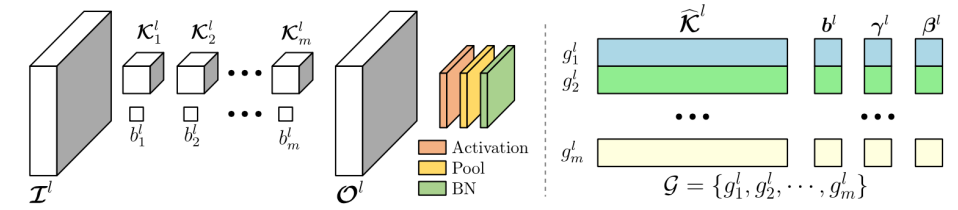
introduce Auto-Train-Once (ATO), illustrated in Figure 4. ATO employs a subset of samples to train a network controller, enabling dynamic management of the pruning operation on zero-invariant groups (ZIGs). Our experimental results affirm the success of the algorithm in effectively identifying the optimal selection of ZIGs through the network controller in a dynamic way.

4.2 Proposed Method

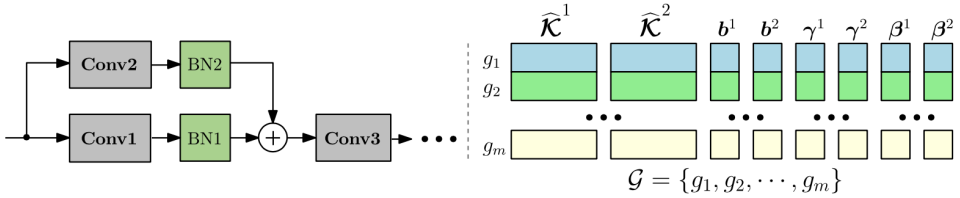
The core concept of the approach involves training a target network under the supervision of a trainable controller network. This controller network assumes the responsibility of generating a mask $\mathbf{w}$ for each group in zero-invariant groups (ZIGs) [16]. Upon completion of the training process, the compression model is formed by directly eliminating elements masked out by $\mathbf{w}$ without requiring further adjustments.

4.2.1 Zero-Invariant Groups

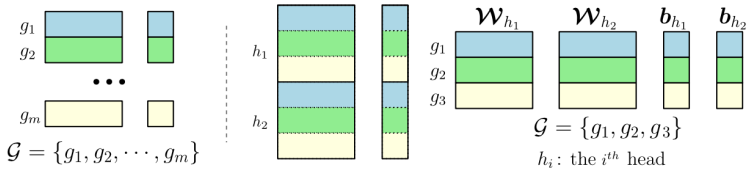
Definition 2. *(Zero-Invariant Groups (ZIGs)) [16]. In the context of a layer-wise Deep Neural Network (DNN), entire trainable parameters are divided into disjoint groups* $\mathcal{G} = \{g\}$. *These groups are termed zero-invariant groups (ZIGs) when each group* $g \in \mathcal{G}$ *exhibits zero-*



(a) Conv-BN. m denotes the number of channels in $\mathcal{O}^l$.



(b) Residual block. m denotes the number of output channels of the residual block.



(c) Fully connected layer (Left). Multi-head attention layer (Right). m denotes the length of output vector.

Figure 5: Zero-invariant group partition for three popular structures [15].

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. [16] initially introduced zero-invariant groups (ZIGs), as depicted in Figure 5. In this case, if all weights in a group is 0, we can directly remove this group and will not change the output. To illustrate, let's consider the structure of a Convolutional layer (Conv) without bias followed by the batch-normalization layer (BN), as outlined below:

$$\mathcal{O}^l \leftarrow \mathcal{I}^l \otimes \hat{\mathcal{K}}^l, \mathcal{I}^{l+1} \leftarrow \frac{a\left(\mathcal{O}^l\right) - \boldsymbol{\mu}^l}{\boldsymbol{\sigma}^l} \odot \boldsymbol{\gamma}^l + \boldsymbol{\beta}^l, \tag{4-1}$$

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

4.2.2 Controller Network

Within the framework of zero-invariant groups (ZIGs), the Controller Network generates group-wise masks denoted as $\mathbf{w} \in \{0,1\}^N$. The binary values 0 and 1 signify the actions of removing and preserving channel groups, respectively. The Controller Network integrates bi-directional gated recurrent units (GRU) [19] followed by linear layers, and Gumbel-Sigmoid [48] combined with a straight-through estimator (STE) [8]. The inclusion of Gumbel-Sigmoid aims to generate a binary vector $\mathbf{w}$ that approximates a binomial distribution. Additional details about the Controller Network can be found in the appendix.

Utilizing the obtained mask $\mathbf{w}$, we can apply it to the feature maps to control the output of each group in ZIGs. For example, in a convolutional neural network (CNN), if the channels of the l^{th} layer are in ZIGs and the weights of the l^{th} layer are represented as $\mathcal{M}_l \in \mathbb{R}^{C_l \times C_{l-1} \times k_l \times k_l}$, where C_l is the number of channels and k_l is the kernel size in the l^{th} layer. The feature map of the l^{th} layer can be denoted by $\mathcal{F}_l \in \mathbb{R}^{C_l \times W_l \times H_l}$, where H_l and W_l are the height and width of the current feature map. With the mask $\mathbf{w}_l = \{0,1\}^{C_l}$ for the

l^{th} layer, the feature map of the l^{th} layer is then modified as $\widehat{\mathcal{F}}_l = \mathbf{w}_l \odot \mathcal{F}_l$. Following the definition of ZIGs, setting the output as 0 for one ZIG is equivalent to setting all weights in this ZIG to 0.

4.2.3 Auto-Train-Once

In this section, we present our proposed algorithm, Auto-Train-Once (ATO). The specifics of ATO are outlined in Algorithm 7.

To commence, we initialize the zero-invariant group set, denoted as $\mathcal{G}$, by partitioning the trainable parameters of $\mathcal{M}$. Following this, we construct a controller network with model weights $\mathcal{W}$ in such a way that the output dimension equals $|\mathcal{G}|$ (where $|\cdot|$ denotes set cardinality). Subsequently, the controller network generates the model mask vector $\mathbf{w} = CN(\mathcal{W})$, and groups in ZIGs $\mathcal{G}$ with a mask value of 0 will incur a penalty in the projection operation, as indicated in Line 8 of Algorithm 7.

We can formulate the optimization problem with regularization as follows:

$$\begin{aligned}\min_{\mathcal{M}} \ \mathcal{J}(\mathcal{M}) :=& \mathcal{L}(\mathcal{M}) + g(\mathcal{M}) \\ =& \mathcal{L}\big(f(x;\mathcal{M}), y\big) + \sum_{g\in\mathcal{G}} \lambda_g \left\|[\mathcal{M}]_g\right\|, \end{aligned} \tag{4-2}$$

where $f(x;\mathcal{M})$ represents the output of the target model with weight $\mathcal{M}$, $\mathcal{L}\big(f(x;\mathcal{M}), y\big)$ denotes the loss function with data (x, y), and $\mathcal{G}$ signifies the zero-invariant groups (ZIGs). The regularization coefficient for each group, λ_g, is determined by the output of the controller network, i.e., $\lambda_g = \lambda(1 - [\mathbf{w}]_g)$. If λ_g is 0, there is no penalty imposed on group g. After T_w warm-up steps, regularization is introduced to prune the target model. A higher value of λ typically leads to increased group sparsity [111].

To integrate group sparsity into optimization objective functions, various existing projection operators, such as the Half-Space Projector (HSP) [16], can be utilized as outlined below:

$$\left[\mathrm{Proj}^{HS}_{\mathcal{S}_k}(\boldsymbol{z})\right]_g := \begin{cases} 0 & \text{if } [\boldsymbol{z}]_g^\top [\mathcal{M}]_g < \epsilon \left\|[\mathcal{M}]_g\right\|^2 \\ [\boldsymbol{z}]_g & \text{otherwise.} \end{cases} \tag{4-3}$$

and proximal gradient projector [111] as follows:.

$$\text{prox}_{\eta\lambda_g}([\boldsymbol{z}]_g) = \begin{cases} [\boldsymbol{z}]_g - \eta\lambda_g \frac{[\boldsymbol{z}]_g}{\|[\boldsymbol{z}]_g\|_2}, \\ \text{ if } \| [\boldsymbol{z}]_g \| \geq \alpha\lambda_g, \\ 0, \text{ otherwise .} \end{cases} \tag{4-4}$$

Conversely, to prevent entrapment in local optima, we employ a controller network trained to dynamically modify the model mask from T_{start} to T_{end}. We utilize a subset of the training dataset $\mathcal{D}$ to create $\mathcal{D}_{\text{CN}}$. The comprehensive loss function for the controller network is as follows:

$$\min_{\mathcal{W}} \mathcal{J}_{\text{CN}}(\mathcal{W}) := \mathcal{L}\big(f(x; \mathcal{M}, \mathbf{w}), y\big) + \gamma \mathcal{R}_{\text{FLOPs}}(P(\mathbf{w}), pP_{\text{total}}), \tag{4-5}$$

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:

$$\mathcal{R}_{\text{FLOPs}}(x, y) = \log(\max(x, y)/y). \tag{4-6}$$

In the Auto-Train-Once (ATO) approach, we iteratively train the target model and controller network. Upon reaching T_{end} epochs, we halt the training of the controller network and fix the model mask vector $\mathbf{w}$ to enhance the stability of model training in the concluding phase.

Algorithm 7 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 η, λ, γ, total steps T, warm-up steps T_{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 Equation (4-2).

    Update model weights $\mathcal{M}$ with any stochastic optimizer.

    if $T \geq T_{\text{w}}$ **then**

      Perform projection operator and update following Equation (4-3) or Equation (4-4) 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 8 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 $\mathcal{D}_{\text{CN}}$, γ

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 Equation (4-5).

  Update the controller network weight $\mathcal{W}$ with stochastic optimizer.

end for

Generate mask $\mathbf{w}$

Output: controller network $\mathcal{M}$ and $\mathbf{w}$

4.3 Convergence and Complexity Analysis

In this section, we offer a theoretical analysis to guarantee the convergence of Auto-Train-Once (ATO) to the solution of Equation (4-2), considering both theoretical and practical aspects. Note: for convenience, we designate z as the vector of network weights $\mathcal{M}$. The details of the proof are provided in the appendix.

Assumption 4 (Unbiased gradient and bounded variance)**.** *Set $\xi = (x, y)$ as a batch of data points, and 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 data points ξ has an unbiased stochastic gradient with bounded variance σ^2, i.e.,*

$$\mathbb{E}[\nabla_z \mathcal{L}(z; \xi)] = \nabla_z \mathcal{L}(z) \tag{4-7}$$

$$\mathbb{E}\|\nabla_z \mathcal{L}(z; \xi) - \nabla_z \mathcal{L}(z)\|^2 \leq \sigma^2 \tag{4-8}$$

Assumption 5 (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*

$$\|\nabla_z \mathcal{L}(z_1) - \nabla_z \mathcal{L}(z_2)\| \leq L\|z_1 - z_2\| \tag{4-9}$$

Assumption 4 and Assumption 5 are standard assumptions in stochastic optimization and are widely used in deep learning convergence analysis [32, 2, 21, 105, 52].

4.3.1 Stochastic Mirror Descent Method

We transform our algorithm (ATO) into the stochastic mirror descent method for the convergence analysis, encompassing both stochastic non-adaptive optimizers (such as SGD) and stochastic adaptive optimizers (like ADAM). It shows that our algorithms and theoretical analysis consider almost all existing popular optimizers.

For convenience, let z denote the vector of network weights $\mathcal{M}$. To solve the general minimization optimization problem $\min_z f(z)$, the mirror descent method [12, 7] follows the below step:

$$z_{t+1} = \arg\min_{z} \left\{ f(z_t) + \langle \nabla f(z_t), z - z_t \rangle + \frac{1}{\eta} D_\phi(z, z_t) \right\}, \tag{4-10}$$

where $\eta > 0$ is learning rate.

The Bregman divergence, i.e., Bregman distance, is defined as follows:

$$D_\phi(z, x) = \phi(z) - \phi(x) - \langle \nabla \phi(x), z - x \rangle \tag{4-11}$$

and we can define $\phi_t(z) = \frac{1}{2} z^T A_t z$.

It is worth noting that the first two terms in the given function Equation (4-10) represent a linear approximation of $f(z)$, while the last term accounts for the Bregman distance between z and z_t. Notably, the constant terms $f(z_t)$ and $\langle \nabla f(z_t), z_t \rangle$ can be disregarded in the function. By choosing $\phi(z) = \frac{1}{2}\|z\|^2$, we obtain $D_\phi(z, z_t) = \frac{1}{2}\|z - z_t\|^2$. Consequently, we arrive at the standard gradient descent algorithm as follows:

$$z_{t+1} = z_t - \eta \nabla f(z_t). \tag{4-12}$$

In practice, the full gradient is computationally expensive, thus a stochastic gradient estimator is used to speed up training. Furthermore, the stochastic mirror descent update step is as below:

$$z_{t+1} = \arg\min_{z} \left\{ \langle m_t, z \rangle + \frac{1}{\eta_t} D_{\phi_t}(z, z_t) + g_t(z) \right\}, \tag{4-13}$$

where m_t is the gradient estimator of $\nabla \mathcal{L}(z; \xi)$ and we can use momentum gradient estimator as $m_t = (1 - \alpha_t) m_{t-1} + \alpha_t \nabla \mathcal{L}(z; \xi)$, η_t is the learning rate, $g(z)$ is a generally nonsmooth

regularization in Equation (4-2). The controller network uses the generated mask to adjust group-specific regularization coefficient λ_g in $g(z)$.

In the practice, since regularization $g(x)$ in composite functions (4-2) might not differentiable, we can minimize the loss function $\mathcal{L}$ firstly as step 6 in Algorithm 7, which is equivalent to the following generalized problem:

$$\tilde{z}_{t+1} = \arg\min_{z} \big\{ \langle m_t, z \rangle + \frac{1}{\gamma} D_t(z, z_t) \big\}, \tag{4-14}$$

and then perform the projection operator as step 8 in Algorithm 7 as in Equation (4-3) and Equation (4-4) 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 [55], defined as

$$\begin{aligned} &\tilde{v}_0 = 0,\ \tilde{v}_t = \beta \tilde{v}_{t-1} + (1-\beta)\nabla_z \mathcal{L}(z_t; \xi_t)^2, \\ &A_t = \mathrm{diag}(\sqrt{\tilde{v}_t} + \epsilon), \end{aligned} \tag{4-15}$$

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

$$D_t(z, z_t) = \frac{1}{2}(z - z_t)^T A_t (z - z_t). \tag{4-16}$$

Table 4: 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 [17]	93.02 %	92.86%	-0.16%	79.7%
		ATO (ours)	94.41%	**94.51**%	+ **0.10**%	**79.8**%
	ResNet-56	DCP-Adapt [119]	93.80%	93.81%	+0.01%	47.0%
		SCP [50]	93.69%	93.23%	−0.46%	51.5%
		FPGM [41]	93.59%	92.93%	−0.66%	52.6%
		SFP [40]	93.59%	92.26%	−1.33%	52.6%
		FPC [39]	93.59%	93.24%	−0.25%	52.9%
		HRank [73]	93.26%	92.17%	−0.09%	50.0%
		DMC [31]	93.62%	92.69%	+0.07%	50.0%
		GNN-RL [110]	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 [119]	94.47%	94.17%	−0.30%	26.0%
		DCP [119]	94.47%	94.69%	+0.22%	26.0%
		DMC [31]	94.23%	94.49%	+0.26%	40.0%
		SCOP [96]	94.48%	94.24%	-0.24%	40.3%
		ATO (ours)	94.45%	**94.78%**	**+0.33%**	**45.8%**
CIFAR-100	ResNet-18	OTOv2 [17]	-	74.96%	-	39.8%
		ATO (ours)	77.95%	**76.79**%	**−0.07**%	**40.1**%
	ResNet-34	OTOv2 [17]	-	76.31%	-	49.5%
		ATO (ours)	78.43 %	**78.54** %	**+0.11**%	**49.5**%

4.3.2 Convergence Metrics and analysis

We introduce useful convergence metrics to measure the convergence of our algorithms. As in [32], we define a generalized projected gradient gradient mapping as:

$$\mathcal{P}_t = \frac{1}{\eta_t}(z_t - z_{t+1}^*), \tag{4-17}$$
$$z_{t+1}^* = \arg\min_z \left\{ \langle \nabla \mathcal{L}(z_t), z\rangle + \frac{1}{\eta_t} D_{\phi_t}(z, z_t) + g(z) \right\}$$

Therefore, for Problem (4-2), 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 Assumption 4 and Assumption 5. The following theorems show our main theoretical results. All related proofs are provided in the Supplement Material.

Theorem 3. *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*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathcal{P}_t\| \leq \frac{\sqrt{G}\bar{c}^{1/4}}{T^{1/2}} + \frac{\sqrt{G}}{T^{1/4}} \tag{4-18}$$

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

Remark 6. *(Complexity) To make the* $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\,\|\mathcal{P}_t\| \leq \varepsilon$, *we get* $T = O(\varepsilon^{-4})$. *Since we use the constant batch size,* $b = O(1)$, *we get complexity* $bT = O(\varepsilon^{-4})$, *aligns with the standard complexity for stochastic optimizers [32] and guarantees the convergence of the proposed algorithm.*

4.3.3 Related Proof

In this subsection, we provide a detailed proof 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 9, to facilitate analysis. Since the mask is dynamic and the objective function varies before T_{end}, we mainly analyze the convergence after the mask is fixed and assume the point will not be far away from the optimal point during the first phase of training. It should be noted that t in Algorithm 9 denotes the update iteration steps to discuss the training progression in each step, instead of the training epoch index as before.

As discussed before, we define a Bregman distance [11, 12, 32] associated with $\phi(z)$ as follows:

$$D_\phi(z', z) = \phi(z') - \big[\phi(z) + \langle\nabla\phi(z), z' - z\rangle\big], \quad \forall z, z'. \tag{4-19}$$

Given that $\phi(z) = \frac{1}{2}z^T Az$ in our dissertation, 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 [55], defined as

$$\begin{aligned} &\tilde{v}_0 = 0,\ \tilde{v}_t = \beta\tilde{v}_{t-1} + (1-\beta)\nabla_z\mathcal{L}(z_t;\xi_t)^2, \\ &A_t = \mathrm{diag}(\sqrt{\tilde{v}_t} + \epsilon), \end{aligned} \tag{4-20}$$

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 lemmas.

Lemma 7. *(Lemma 1 in [32]) 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*

$$\langle \nabla_z\mathcal{L}(z), \mathcal{P}_t\rangle \geq \epsilon\|\mathcal{P}_t\|^2 + \frac{1}{\eta}\big[g(z_{t+1}^+) - g(z_t)\big], \tag{4-21}$$

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

Based on Lemma 7, 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

$$\langle m_t, \tilde{\mathcal{P}}_t\rangle \geq \epsilon\|\tilde{\mathcal{P}}_t\|^2 + \frac{1}{\eta}(g(z_{t+1}) - g(x_t)). \tag{4-22}$$

Lemma 8. *Assume that the stochastic partial derivatives m_{t+1} be generated from Algorithm 9, we have*

$$\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - m_{t+1}\|^2 \leq (1-\alpha_{t+1})\mathbb{E}\|\nabla_z\mathcal{L}(z_t)) - m_t\|^2 + \frac{L^2\eta_t^2}{\alpha_{t+1}}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\alpha_{t+1}^2\sigma^2}{b} \tag{4-23}$$

Algorithm 9 ATO Algorithm (Mirror Descant)

Input: Target model with model weights vector z (no need to be pre-tained). Datasets D, D_{CN}, γ, λ, η, total training iteration T and each epoch of $\mathcal{D}$ has Q iterations; controller network training steps T_{start} and T_{end}.

for $t = 1, 2 \ldots, T$ **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 $z_{t+1} = \arg\min_z \big\{\langle m_t, z\rangle + \frac{1}{\eta_t} D_{\phi_t}(z, z_t) + g(z)\big\}$;

  if $T_{\text{start}}Q \le T \le T_{\text{end}}Q$ **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

$$
\begin{aligned}
&\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - m_{t+1}\|^2 = \mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - \alpha_{t+1}\nabla_z\mathcal{L}(z_{t+1}; \xi_{t+1}) - (1 - \alpha_{t+1})m_t\|^2 \\
=&\mathbb{E}\|\alpha_{t+1}(\nabla_z\mathcal{L}(z_{t+1}) - \nabla_z\mathcal{L}(z_{t+1}; \xi_{t+1})) + (1 - \alpha_{t+1})(\nabla_z\mathcal{L}(z_t) - m_t) \\
&\quad + (1 - \alpha_{t+1})\big(\nabla_z\mathcal{L}(z_{t+1}) - \nabla_z\mathcal{L}(z_t)\big)\|^2 \\
\overset{(a)}{=}&\mathbb{E}\|(1 - \alpha_{t+1})(\nabla_z\mathcal{L}(z_t) - m_t) + (1 - \alpha_{t+1})\big(\nabla_z\mathcal{L}(z_{t+1}) - \nabla_z\mathcal{L}(z_t)\big)\|^2 \\
&\quad + \alpha_{t+1}^2\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - \nabla_z\mathcal{L}(z_{t+1}; \xi_{t+1})\|^2 \\
\le&(1 - \alpha_{t+1})^2(1 + \frac{1}{\alpha_{t+1}})\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - \nabla_z\mathcal{L}(z_t)\|^2 \\
&\quad + (1 - \alpha_{t+1})^2(1 + \alpha_{t+1})\mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 + \alpha_{t+1}^2\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - \nabla_z\mathcal{L}(z_{t+1}; \xi_{t+1})\|^2 \\
\overset{(b)}{\le}&(1 - \alpha_{t+1})\mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 + \frac{1}{\alpha_{t+1}}\mathbb{E}\|\nabla_z\mathcal{L}(z_{t+1}) - \nabla_z\mathcal{L}(z_t))\|^2 + \frac{\alpha_{t+1}^2\sigma^2}{b} \\
\le&(1 - \alpha_{t+1})\mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 + \frac{L^2}{\alpha_{t+1}}\mathbb{E}\|z_{t+1} - z_t\|^2 + \frac{\alpha_{t+1}^2\sigma^2}{b} \\
=&(1 - \alpha_{t+1})\mathbb{E}\|\nabla_z\mathcal{L}(z_t) - m_t\|^2 + \frac{L^2\eta_t^2}{\alpha_{t+1}}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\alpha_{t+1}^2\sigma^2}{b},
\end{aligned}
\tag{4-24}
$$

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 $0 \le \alpha_{t+1} \le 1$ such that $(1 - \alpha_{t+1})^2(1 + \alpha_{t+1}) = 1 - \alpha_{t+1} - \alpha_{t+1}^2 + \alpha_{t+1}^3 \le 1 - \alpha_{t+1}$ and $(1 - \alpha_{t+1})^2(1 + \frac{1}{\alpha_{t+1}}) \le$

$(1-\alpha_{t+1})(1+\frac{1}{\alpha_{t+1}}) = -\alpha_{t+1} + \frac{1}{\alpha_{t+1}} \leq \frac{1}{\alpha_{t+1}}$, and the b is the batch size; the last inequality holds by the definition of $\tilde{\mathcal{P}}_t$ in Equation (4-22). □

Lemma 9. *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

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

we have

$$\|\nabla \mathcal{L}(z_t) - m_t\| \geq \epsilon \left\| \mathcal{P}_t - \tilde{\mathcal{P}}_t \right\| \tag{4-25}$$

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

$$\left\langle m_t + \nabla g(z_{t+1}) + \frac{1}{\eta_t} \left(\nabla \mathcal{L}_t(z_{t+1}) - \nabla \mathcal{L}_t(z_t) \right), z - z_{t+1} \right\rangle \geq 0 \tag{4-26}$$

$$\left\langle \nabla \mathcal{L}(z_t) + \nabla g\left(z_{t+1}^{+}\right) + \frac{1}{\eta_t} \left(\nabla \mathcal{L}_t\left(z_{t+1}^{+}\right) - \nabla \mathcal{L}_t(z_t) \right), z - z_{t+1}^{+} \right\rangle \geq 0 \tag{4-27}$$

where $\nabla g(z_{t+1}) \in \partial g(z_{t+1})$. Taking $z = z_{t+1}^{+}$in the Equation (4-26) and $z = z_{t+1}$ in the Equation (4-27), by the convexity of $g(x)$, we have

$$\begin{aligned} \left\langle m_t, z_{t+1}^{+} - z_{t+1} \right\rangle &\geq \left\langle \nabla g(z_{t+1}), z_{t+1} - z_{t+1}^{+} \right\rangle + \frac{1}{\eta_t} \left\langle \nabla \mathcal{L}_t(z_{t+1}) - \nabla \mathcal{L}_t(z_t), z_{t+1} - z_{t+1}^{+} \right\rangle \\ &\geq g(z_{t+1}) - g\left(z_{t+1}^{+}\right) + \frac{1}{\eta_t} \left\langle \nabla \mathcal{L}_t(z_{t+1}) - \nabla \mathcal{L}_t(z_t), z_{t+1} - z_{t+1}^{+} \right\rangle \end{aligned} \tag{4-28}$$

$$\begin{aligned} \left\langle \nabla \mathcal{L}(z_t), z_{t+1} - z_{t+1}^{+} \right\rangle &\geq \left\langle \nabla g\left(z_{t+1}^{+}\right), z_{t+1}^{+} - z_{t+1} \right\rangle + \frac{1}{\eta_t} \left\langle \nabla \mathcal{L}_t\left(z_{t+1}^{+}\right) - \nabla \mathcal{L}_t(z_t), z_{t+1}^{+} - z_{t+1} \right\rangle \\ &\geq g\left(z_{t+1}^{+}\right) - g(z_{t+1}) + \frac{1}{\eta_t} \left\langle \nabla \mathcal{L}_t\left(z_{t+1}^{+}\right) - \nabla \mathcal{L}_t(z_t), z_{t+1}^{+} - z_{t+1} \right\rangle \end{aligned} \tag{4-29}$$

Summing up the above inequalities, we obtain

$$\begin{aligned}&\left\langle \nabla\mathcal{L}\left(z_t\right)-m_t, z_{t+1}-z_{t+1}^{+}\right\rangle\\ \geq&\frac{1}{\eta_t}\left\langle \nabla\mathcal{L}_t\left(z_{t+1}^{+}\right)-\nabla\mathcal{L}_t\left(z_{t+1}\right), z_{t+1}^{+}-z_{t+1}\right\rangle \geq \frac{\epsilon}{\eta_t}\left\|z_{t+1}^{+}-z_{t+1}\right\|^2\end{aligned} \tag{4-30}$$

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 $\left\|\mathcal{P}_t-\tilde{\mathcal{P}}_t\right\| = \|\frac{1}{\eta_t}\left(z_t-z_{t+1}^{+}\right)-\frac{1}{\eta_t}\left(z_t-z_{t+1}\right)\|=\frac{1}{\eta_t}\left\| z_{t+1}^{+}-z_{t+1}\right\|$, we have

$$\left\|\nabla\mathcal{L}\left(z_t\right)-m_t\right\| \geq \epsilon\left\|\mathcal{P}_t-\tilde{\mathcal{P}}_t\right\| \tag{4-31}$$

□

Lemma 10. *Suppose the sequence* $\{z_t\}_{t=1}^{T}$ *be generated from Algorithms 9. Let* $0<\eta_t\leq \min\{1,\frac{\epsilon}{4L}\}$*, we have*

$$\mathcal{J}\left(z_{t+1}\right) \leq \mathcal{J}\left(z_t\right)-\frac{\eta_t\epsilon}{4}\left\|\mathcal{P}_t\right\|^2+\frac{2\eta_t}{\epsilon}\left\|m_t-\nabla_z\mathcal{L}\left(z_t\right)\right\|^2 \tag{4-32}$$

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

$$\begin{aligned}\mathcal{L}\left(z_{t+1}\right) &\leq \mathcal{L}\left(z_t\right)+\left\langle \nabla\mathcal{L}\left(z_t\right), z_{t+1}-z_t\right\rangle+\frac{L}{2}\left\|z_{t+1}-z_t\right\|^2\\ &=\mathcal{L}\left(z_t\right)-\eta_t\left\langle \nabla\mathcal{L}\left(z_t\right), \tilde{\mathcal{P}}_t\right\rangle+\frac{\eta_t^2L}{2}\left\|\tilde{\mathcal{P}}_t\right\|^2\\ &=\mathcal{L}\left(z_t\right)-\eta_t\left\langle m_t, \tilde{\mathcal{P}}t\right\rangle+\eta_t\left\langle m_t-\nabla\mathcal{L}\left(z_t\right), \tilde{\mathcal{P}}_t\right\rangle+\frac{\eta_t^2L}{2}\left\|\tilde{\mathcal{P}}_t\right\|^2\\ &\overset{(a)}{\leq}\mathcal{L}\left(z_t\right)-\eta_t\epsilon\left\|\tilde{\mathcal{P}}_t\right\|^2-g\left(z_{t+1}\right)+g\left(z_t\right)+\eta_t\left\langle m_t-\nabla\mathcal{L}\left(z_t\right), \tilde{\mathcal{P}}_t\right\rangle+\frac{\eta_t^2L}{2}\left\|\tilde{\mathcal{P}}_t\right\|^2\\ &\overset{(b)}{\leq}\mathcal{L}\left(z_t\right)+\left(\frac{\eta_t^2L}{2}-\frac{3\eta_t\epsilon}{4}\right)\left\|\tilde{\mathcal{P}}_t\right\|^2-g\left(z_{t+1}\right)+g\left(z_t\right)+\frac{\eta_t}{\epsilon}\left\|m_t-\nabla\mathcal{L}\left(z_t\right)\right\|^2\end{aligned} \tag{4-33}$$

where the (a)holds by the above Lemma 7, and the (b) holds by the following Cauchy-Schwarz inequality and Young's inequality as

$$\begin{aligned}\left\langle m_t-\nabla\mathcal{L}\left(z_t\right), \tilde{\mathcal{P}}_t\right\rangle &\leq \left\|m_t-\nabla\mathcal{L}\left(z_t\right)\right\|\left\|\tilde{\mathcal{P}}_t\right\|\\ &\leq \frac{1}{\epsilon}\left\|m_t-\nabla\mathcal{L}\left(z_t\right)\right\|^2+\frac{\epsilon}{4}\left\|\tilde{\mathcal{P}}_t\right\|^2\end{aligned} \tag{4-34}$$

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

$$\begin{aligned}\mathcal{J}\left(z_{t+1}\right) &\leq \mathcal{J}\left(z_t\right) + \left(\frac{\eta_t^2 L}{2} - \frac{3\eta_t \epsilon}{4}\right)\left\|\tilde{\mathcal{P}}_t\right\|^2 + \frac{\eta_t}{\epsilon}\left\|m_t - \nabla_z \mathcal{L}\left(z_t\right)\right\|^2 \\ &\leq \mathcal{J}\left(z_t\right) - \frac{5\eta_t \epsilon}{8}\left\|\tilde{\mathcal{P}}_t\right\|^2 + \frac{\eta_t}{\epsilon}\left\|m_t - \nabla_z \mathcal{L}\left(z_t\right)\right\|^2, \end{aligned} \tag{4-35}$$

where the last inequality is due to $0 < \eta_t \leq \frac{\epsilon}{4L}$. Based on Lemma 9, we have

$$\left\|\mathcal{P}_t\right\|^2 \leq 2\left\|\tilde{\mathcal{P}}_t\right\|^2 + 2\left\|\tilde{\mathcal{P}}_t - \mathcal{P}_t\right\|^2 \leq 2\left\|\tilde{\mathcal{P}}_t\right\|^2 + \frac{2}{\epsilon^2}\left\|m_t - \nabla \mathcal{L}\left(z_t\right)\right\|^2. \tag{4-36}$$

Finally, we have

$$\mathcal{J}\left(z_{t+1}\right) \leq \mathcal{J}\left(z_t\right) - \frac{\eta_t \epsilon}{8}\left\|\tilde{\mathcal{P}}_t\right\|^2 - \frac{\eta_t \epsilon}{4}\left\|\mathcal{P}_t\right\|^2 + \frac{2\eta_t}{\epsilon}\left\|m_t - \nabla_z \mathcal{L}\left(z_t\right)\right\|^2. \tag{4-37}$$

□

Theorem 4. *(Restatement of Theorem 3) Assume that the sequence $\{z_t\}_{t=1}^T$ be generated from the Algorithm 9. When we have $\eta_t = \frac{\hat{c}}{(\bar{c}+t)^{1/2}}$, $\frac{\hat{c}}{\bar{c}^{1/2}} \leq \min\{1, \frac{\epsilon}{4L}\}$, $c_1 = \frac{4L}{\epsilon}, \alpha_{t+1} = c_1\eta_t$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathcal{P}_t\| \leq \frac{\sqrt{G}\bar{c}^{1/4}}{T^{1/2}} + \frac{\sqrt{G}}{T^{1/4}} \tag{4-38}$$

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

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

According to Lemma 8, we have

$$\begin{aligned}&\mathbb{E}\|\nabla_z \mathcal{L}(z_{t+1}) - m_{t+1}\|^2 - \mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 \\ \leq& -\alpha_{t+1}\mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 + L^2\eta_t^2/\alpha_{t+1}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\alpha_{t+1}^2\sigma^2}{b} \\ =& -c_1\eta_t\mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 + L^2\eta_t/c_1\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{c_1^2\eta_t^2\sigma^2}{b} \\ \leq& -\frac{4L\eta_t}{\epsilon}\mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 + \frac{\epsilon L\eta_t}{4}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\bar{c}\eta_t^2\sigma^2}{\hat{c}^2 q}, \end{aligned} \tag{4-39}$$

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 10, we have

$$\mathcal{J}\left(z_{t+1}\right) \leq \mathcal{J}\left(z_t\right) - \frac{\eta_t \epsilon}{8}\left\|\tilde{\mathcal{P}}_t\right\|^2 - \frac{\eta_t \epsilon}{4}\left\|\mathcal{P}_t\right\|^2 + \frac{2\eta_t}{\epsilon}\left\|m_t - \nabla_z \mathcal{L}\left(z_t\right)\right\|^2 \tag{4-40}$$

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

$$\Omega_t = \mathbb{E}\big[\mathcal{J}(z_t) + \frac{1}{2L}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2\big] \tag{4-41}$$

Then we have

$$\begin{aligned}
&\Omega_{t+1} - \Omega_t = \mathbb{E}\big[\mathcal{J}(z_{t+1}) - \mathcal{J}(z_t)\big] + \frac{1}{2}[\mathbb{E}\|\nabla_z \mathcal{L}(z_{t+1}) - m_{t+1}\|^2 - \mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2] \\
\leq &- \frac{\eta_t \epsilon}{8}\left\|\tilde{\mathcal{P}}_t\right\|^2 - \frac{\eta_t \epsilon}{4}\left\|\mathcal{P}_t\right\|^2 + \frac{2\eta_t}{\epsilon}\left\|m_t - \nabla_z \mathcal{L}\left(z_t\right)\right\|^2 \\
&+ \frac{1}{2L}\bigg(- \frac{4L\eta_t}{\epsilon}\mathbb{E}\|\nabla_z \mathcal{L}(z_t) - m_t\|^2 + \frac{\epsilon L \eta_t}{4}\mathbb{E}\|\tilde{\mathcal{P}}_t\|^2 + \frac{\bar{c}\eta_t^2 \sigma^2}{\hat{c}^2 q}\bigg) \\
\leq &- \frac{\epsilon \eta_t}{4}\mathbb{E}\|\mathcal{P}_t\|^2 + \frac{\bar{c}\sigma^2}{2\hat{c}^2 L q}\eta_t^2.
\end{aligned} \tag{4-42}$$

Then we have

$$\eta_t \mathbb{E}\|\mathcal{P}_t\|^2 \leq \frac{4(\Omega_t - \Omega_{t+1})}{\epsilon} + \frac{2\bar{c}\sigma^2}{\epsilon \hat{c}^2 L b}\eta_t^2. \tag{4-43}$$

Taking average over $t = 1, 2, \cdots, T$ on both sides of (4-43), we have

$$\frac{1}{T}\sum_{t=1}^{T}\eta_t \mathbb{E}\|\mathcal{P}_t\|^2 \leq \sum_{t=1}^{T}\frac{4(\Omega_t - \Omega_{t+1})}{T\epsilon} + \frac{1}{T}\sum_{t=1}^{T}\frac{2\bar{c}\sigma^2}{\epsilon L \hat{c}^2 b}\eta_t^2. \tag{4-44}$$

In addition, we have

$$\Omega_1 = \mathcal{J}(z_1) + \frac{1}{2L}\mathbb{E}\|\nabla_z \mathcal{L}(z_1) - v_1\|^2 \leq \mathcal{J}(z_1) + \frac{\sigma^2}{2bL}, \tag{4-45}$$

where the above inequality holds by Assumption 4. 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

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathcal{P}_t\|^2 &\leq \frac{4}{T\epsilon\eta_T}\sum_{t=1}^{T}\big(\Omega_t - \Omega_{t+1}\big) + \frac{1}{T\eta_T}\sum_{t=1}^{T}\frac{2\bar{c}\sigma^2}{\epsilon L\hat{c}^2 b}\eta_t^2 \\
&\leq \frac{4}{T\epsilon\eta_T}\big(\mathcal{J}(z_1) + \frac{\sigma^2}{2bL} - \mathcal{J}(z^*)\big) + \frac{1}{T\eta_T}\sum_{t=1}^{T}\frac{2\bar{c}\sigma^2}{\epsilon L\hat{c}^2 b}\eta_t^2 \\
&\leq \frac{4(\mathcal{J}(z_1) - \mathcal{J}(z^*))}{T\epsilon\eta_T} + \frac{2\sigma^2}{bL\epsilon\eta_T T} + \frac{2\bar{c}\sigma^2}{\eta_T T\epsilon L\hat{c}^2 b}\int_1^T \frac{\hat{c}^2}{\bar{c}+t}dt \\
&\leq \frac{4(\mathcal{J}(z_1) - \mathcal{J}(z^*))}{T\epsilon\eta_T} + \frac{2\sigma^2}{bL\epsilon\eta_T T} + \frac{2\bar{c}\sigma^2}{\eta_T T\epsilon Lb}\ln(\bar{c}+T) \\
&= \left(\frac{4(\mathcal{J}(z_1) - \mathcal{J}(z^*))}{\epsilon\hat{c}} + \frac{2\sigma^2}{bL\epsilon\hat{c}} + \frac{2\bar{c}\sigma^2}{\hat{c}\epsilon Lb}\ln(\bar{c}+T)\right)\frac{(\bar{c}+T)^{1/2}}{T}, \qquad (4\text{-}46)
\end{aligned}$$

where the second inequality holds by the above inequality (4-45) 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

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[|\mathcal{P}_t\|^2\big] \leq \frac{G}{T}(\bar{c}+T)^{1/2}. \qquad (4\text{-}47)$$

According to Jensen's inequality, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\mathcal{P}_t\|\big] \leq \left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\mathcal{P}_t\|^2\big]\right)^{1/2} \leq \frac{\sqrt{G}}{T^{1/2}}(\bar{c}+T)^{1/4} \leq \frac{\sqrt{G}\bar{c}^{1/4}}{T^{1/2}} + \frac{\sqrt{G}}{T^{1/4}} \qquad (4\text{-}48)$$

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

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathcal{P}_t\| \leq \frac{\sqrt{G}\bar{c}^{1/4}}{T^{1/2}} + \frac{\sqrt{G}}{T^{1/4}}. \qquad (4\text{-}49)$$

Based on the above Equation, we can get the final model to converge to the optimal point. To make $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathcal{P}_t\| \leq \varepsilon$, iteration complexity $T = O(\varepsilon^{-4})$ and since we do not require a large batch size to avoid diverge the final complexity is $O(\varepsilon^{-4})$. □

4.4 Experiments

4.4.1 Setup

We evaluate the efficacy of our algorithm by conducting assessments on image classification tasks, utilizing datasets such as CIFAR-10 [57], CIFAR-100, and ImageNet [25]. For comparison, we employ ResNet [38] and MobileNet-V2 [89].

For comparative purposes with existing algorithms, we adjust the hyper-parameter p as in Equation (4-5) to determine the final remaining Floating Point Operations (FLOPs). We maintain a consistent setting with γ in Equation (4-5) set to 4.0. The value of λ in Equation 4-2 is set to 10 for different models and datasets. The starting epoch of the controller network T_{start} is approximately 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 the supplementary materials. The choice of T_{start} and T_{end} in general implies that the training of the controller network is both straightforward and robust.

To mitigate the training costs associated with the controller network training, we randomly sample 5% of the original dataset $\mathcal{D}$ to construct $\mathcal{D}_{\text{CN}}$, incurring additional costs of less than 5% of the original training expenses. We employ the ADAM optimizer [55] to train the controller network with an initial learning rate of 0.001. Additionally, we utilize the proximal gradient project with l_2 norm in Equation 4-4.

For the training of the target network, we adhere to standard training procedures for ResNets on CIFAR-10, CIFAR-100, and ImageNet. For MobileNet-V2, we employ the training settings outlined in its original paper [89]. The parameter T_w is set at approximately 20% of the total epochs for all models and datasets. Due to space limitations, detailed information on training can be found in the supplementary materials. Our main point of comparison is OTOv2, which also eliminates the need for additional fine-tuning. Additionally, we provide a list of other pruning algorithms.

Table 5: 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 [41]	73.92%	91.62%	72.63% (−1.29%)	91.08% (−0.54%)	41.1%
	Taylor [82]	73.31%	-	72.83% (−0.48%)	-	24.2%
	DMC [31]	73.30%	91.42%	72.57% (−0.73%)	91.11% (−0.31%)	43.4%
	SCOP [96]	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 [119]	76.01%	92.93%	74.95% (−1.06%)	92.32% (−0.61%)	55.6%
	CCP [86]	76.15%	92.87%	75.21% (−0.94%)	92.42% (−0.45%)	54.1%
	FPGM [41]	76.15%	92.87%	74.83% (−1.32%)	92.32% (−0.55%)	53.5%
	ABCP [74]	76.01%	92.96%	73.86% (−2.15%)	91.69% (−1.27%)	54.3%
	DMC [31]	76.15%	92.87%	75.35% (−0.80%)	92.49% (−0.38%)	55.0%
	Random-Pruning [66]	76.15%	92.87%	75.13% (−1.02%)	92.52% (−0.35%)	51.0%
	DepGraph [28]	76.15%	-	75.83% (−0.32%)	-	51.7%
	DTP [71]	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 [71]	76.13%	-	75.24 % (−0.89%)	-	60.9%
	OTOv2 [17]	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 [71]	76.13%	-	74.26% (−1.87%)	-	67.3%
	OTOv1 [16]	76.13%	92.86%	74.70% (−1.43%)	92.10% (−0.76%)	64.5%
	OTOv2 [17]	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 [89]	71.80%	91.00%	69.80% (−2.00%)	89.60% (−1.40%)	30.0%
	AMC [42]	71.80%	-	70.80% (−1.00%)	-	30.0%
	CC [70]	71.88%	-	70.91% (−0.97%)	-	28.3%
	MetaPruning [76]	72.00%	-	71.20% (−0.80%)	-	**30.7%**
	Random-Pruning [66]	71.88%	-	70.87% (−1.01%)	-	29.1%
	ATO (ours)	71.88%	90.29%	**72.02%** (**+0.14%**)	**90.19%** (**−0.10%**)	30.1%

4.4.2 CIFAR-10

For CIFAR-10, we opt for ResNet-18, ResNet-56, and MobileNetV2 as our target models. Table 4 displays the outcomes of our algorithm (ATO) and other baseline methods on CIFAR-10.

ResNet-18: Regarding ResNet-18, our algorithm outperforms other methods, achieving the best performance (measured by Δ-Acc) compared to OTOv2 at the same pruned FLOPs. OTOv2 experiences a decline of 0.16% in top-1 accuracy as it avoids the fine-tuning stage and relies on static pruning groups. Guided by the controller network, our algorithm can select masks more precisely, overcoming the limitations of OTOv2 and achieving superior

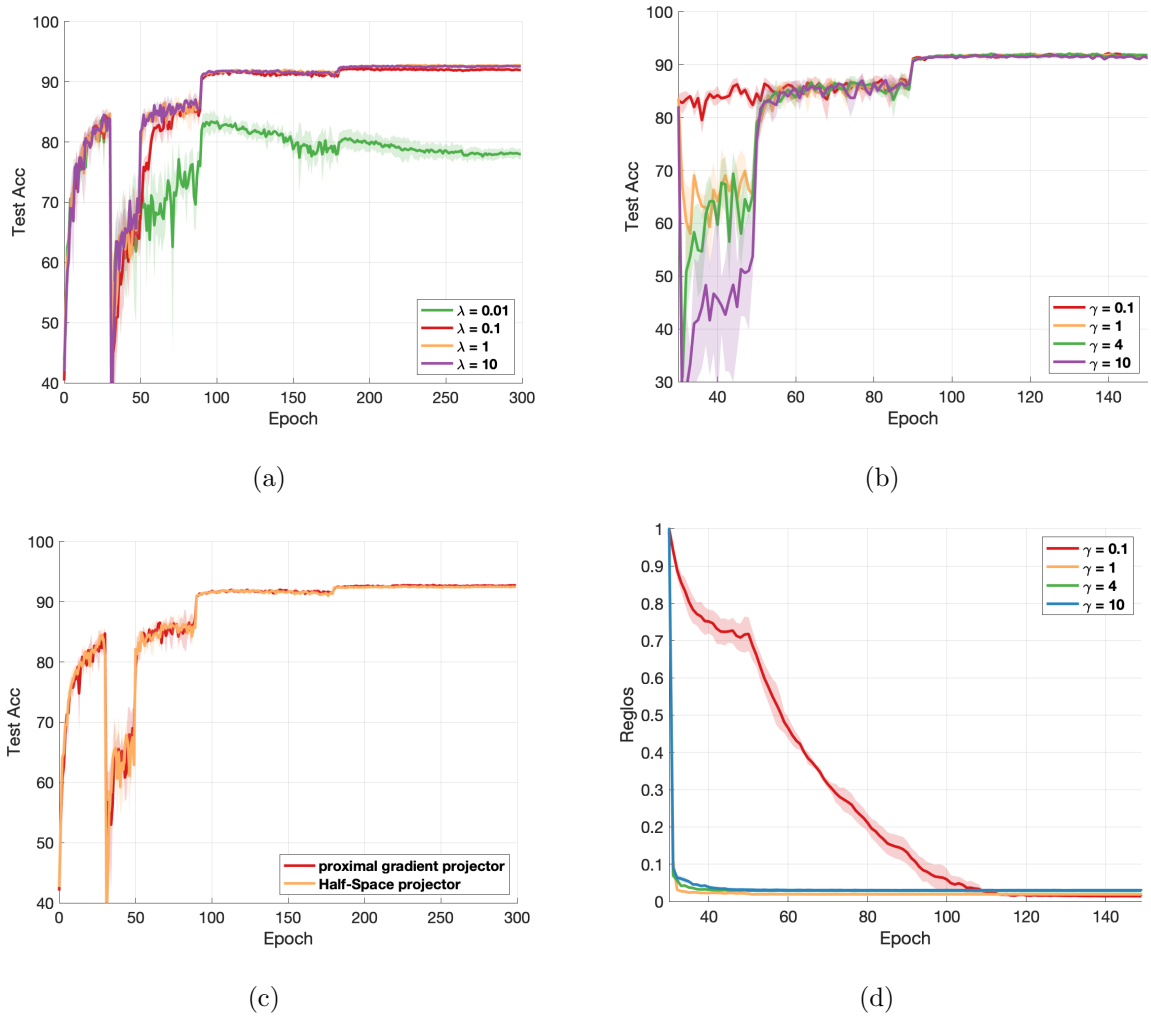


Figure 6: (a): the impact of λ in regularization term in Equation (4-2). (b): the effect of hyperparameter γ in $\mathcal{R}_{\text{FLOPs}}$ in Equation (4-5). (c): the effect of T_{w}. (d): the effect of the project operation as in Equation (4-3). Experiments are conducted on CIFAR-10 with ResNet-56 and $p = 0.45$.

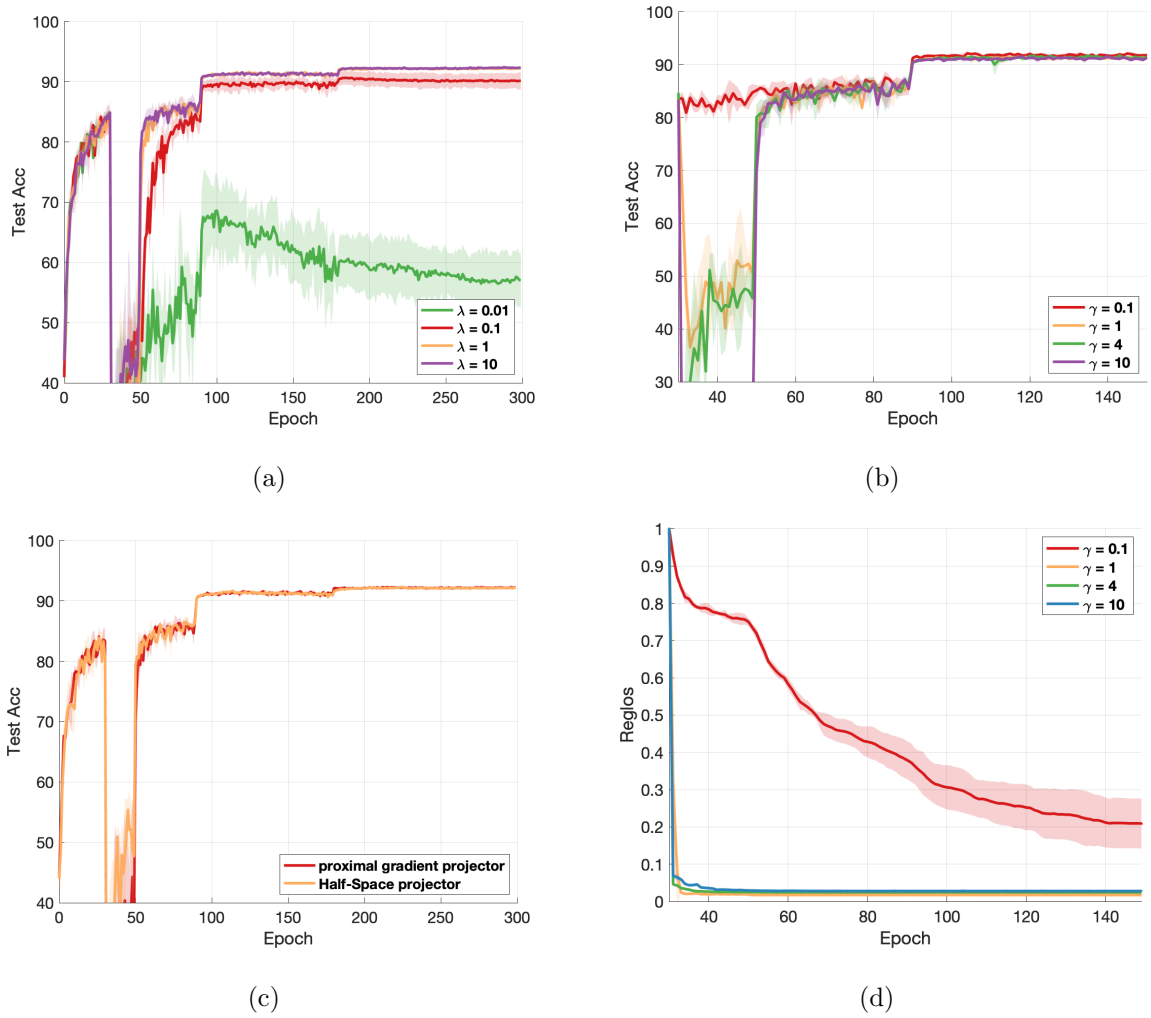


Figure 7: (a): the impact of λ in regularization term in Equation (4-2). (b): the effect of hyperparameter γ in $\mathcal{R}_{\text{FLOPs}}$ in Equation (4-5). (c): the effect of T_{w}. (d): the effect of the project operation as in Equation (4-3). Experiments are conducted on CIFAR-10 with ResNet-56 and $p = 0.35$.

performance.

ResNet-56: For ResNet-56, our method demonstrates superior performance compared to baselines with similar pruned FLOPs. Notably, our approach does not depend on a fine-tuning stage, making it simpler and more user-friendly. Furthermore, our algorithm surpasses the second-best method, GNN-RL, by 0.14% according to Δ-Acc (ATO +0.24% vs. GNN-RL +0.10%) when pruning a slightly larger percentage of FLOPs (ATO 55.0% vs. GNN-RL 54.0%). The performance gaps between our algorithm and others are even more pronounced.

MobileNet-V2: In MobileNet-V2, our method exhibits strong performance, pruning a significant proportion of FLOPs (45.8%) and achieving the best performance in terms of Δ-ACC (+0.33).

4.4.3 CIFAR-100

In our CIFAR-100 comparisons, we focus on ResNet-18 and ResNet-34. The results for the CIFAR-100 dataset are presented in Table 4.

Since OTOv2 does not provide results for CIFAR-100 with ResNet-18 and ResNet-34, we generate OTOv2 results on CIFAR-100 under the same settings as ours. In comparison with OTOv2 results in Table 4, our algorithm pruned slightly more FLOPs, while significantly improving the overall results.

4.4.4 ImageNet

Following that, we apply ATO to ImageNet to showcase its effectiveness. In this scenario, we select ResNet-34, ResNet-50, and MobileNetV2 as target models. The comparison between existing algorithms and ATO is presented in Table 5.

ResNet-34: In ResNet-34, our algorithm attains superior performance compared to others under similar pruned FLOPs, despite having a simpler training procedure. Our algorithm achieves a Top-1 accuracy of 72.92% and a Top-5 accuracy of 91.15%, surpassing other algorithms. SCOP and DMC prune a comparable number of FLOPs to our algorithm and share the same baseline results. However, our method achieves a pruned Top-1 Accuracy

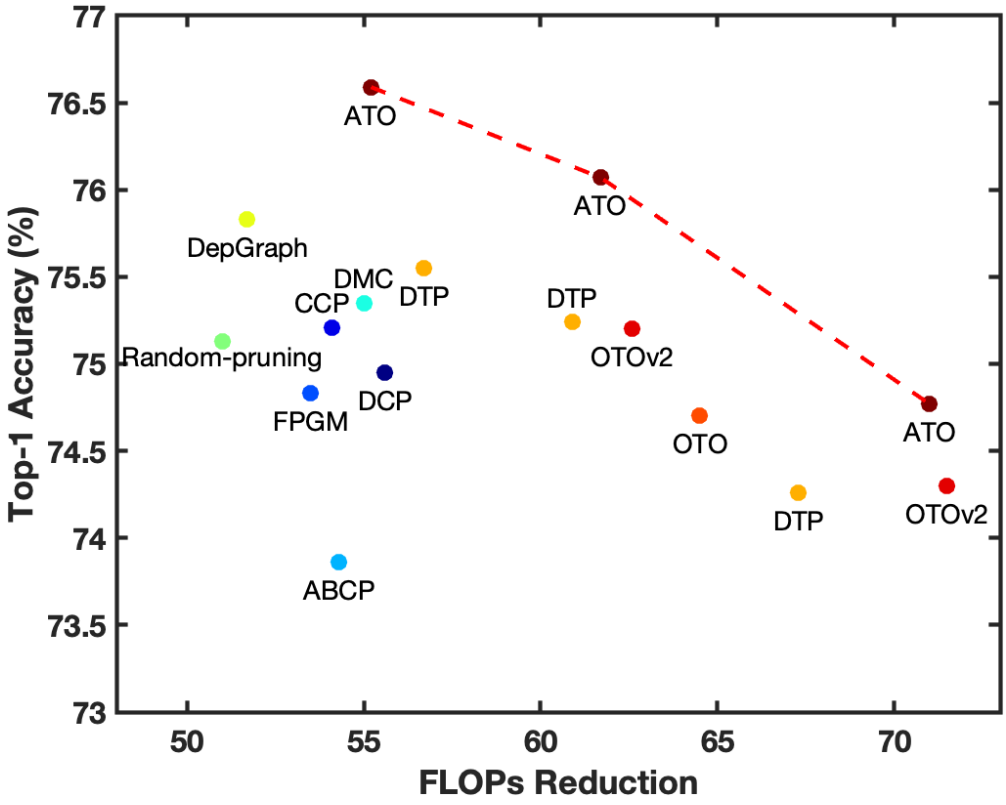


Figure 8: ResNet50 on ImageNet.

that is 0.30% and 0.35% higher than 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.

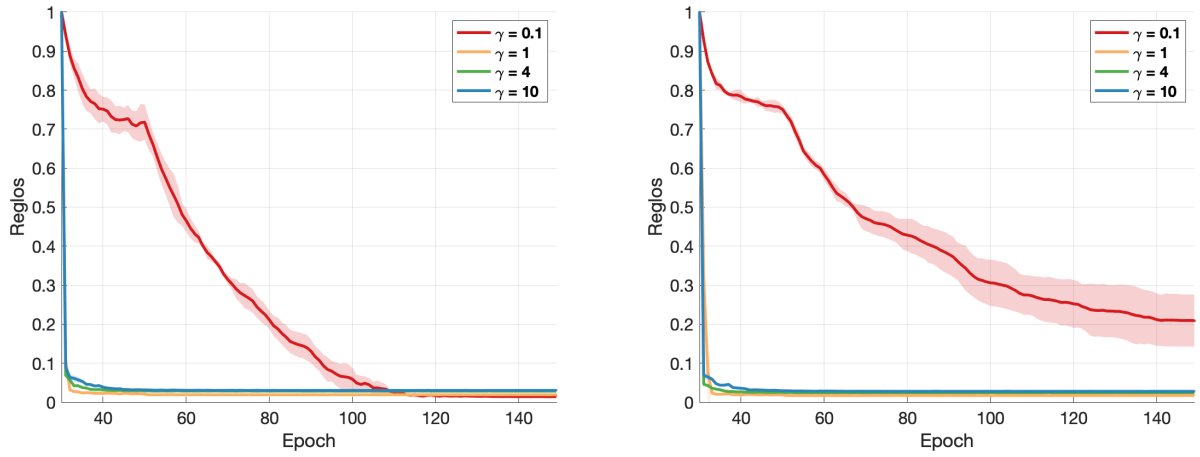
ResNet-50: In ResNet-50, we present a performance portfolio under various pruned FLOPs, ranging from 55.2% to 71.0%. Figure 8 illustrates the comparison with other approaches. While an increase in pruned FLOPs and parameter reductions typically leads to a compromise in accuracy, ATO maintains a leading edge in terms of top-1 accuracy across various levels of FLOPs reduction. Compared with OTOv2, the results of our algorithm do not compromise training simplicity. Notably, under pruned FLOPs of 61.7%, the Top-1 accuracy of ATO reaches 76.07%, surpassing OTOv2 by 0.87% under similar pruned FLOPs. Additionally, even when pruned FLOPs exceed 70%, ATO continues to exhibit strong performance compared to counterparts.

MobileNetV2: The lightweight model MobileNetV2 is generally challenging to compress. Under the pruned FLOPs of 30%, our algorithm achieves the best Top-1 accuracy compared to other methods, even with simpler training procedures. Our algorithm attains a Top-1 accuracy of 72.02% and a Top-5 accuracy of 90.19%, while the results of other counterparts fall below the baseline results.

4.4.5 Ablation Study

We conduct an ablation study to examine the impact of different hyperparameters on model performance and consider different FLOAPs reduction ($p = 0.45$ and $p = 0.35$). For this analysis, we use ResNet-56 on CIFAR-10. It is important to note that the drop in performance at Epoch 30 is due to the initiation of controller network training at that epoch. Subsequently, we evaluate model performance under the mask vector $\mathbf{w}$, which is equivalent to removing the corresponding Zero-Invariant Groups (ZIGs). It should be noted that we plot the standard deviation in shaded errors in Figures 6 and 7, which shows that the range of training fluctuations is small when the model converges. It verifies the robustness and efficiency of our algorithm (ATO).

Impact of λ: We investigate the influence of the regularization coefficient λ in Equation



(a) (b)

Figure 9: the effect of hyperparameter γ in $\mathcal{R}_{\text{FLOPs}}$ in Equation (4-5). Experiments are conducted on CIFAR-10 with ResNet-56 with $p = 0.45$ (a) and $p = 0.35$ (b).

(4-2) and plot the test accuracy in Figures 6 (a) and 7 (a). The curves illustrate that λ plays a crucial role in model training. If the value of λ is too small, it can adversely affect model performance, especially when pruning a large number of FLOPs ($p = 0.35$). However, when it reaches a specific threshold, the effect will vanish.

Impact of γ: We examine the impact of the hyperparameter γ, which controls the strength of FLOPs regularization in Equation (4-5), and plot the test accuracy in Figures 6 and 7. The results indicate that accuracy is minimally affected by γ, and the curves of different γ converge rapidly after the model commences training (Epoch = 30). Additionally, Figure 9 depicts $\mathcal{R}_{\text{FLOPs}}$, showing that the controller network under different γ converges at varying speeds before training concludes at Epoch = 150. It is worth noting that the loss values are scaled to [0, 1]. This suggests 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.

Impact of T_w: We explore the impact of T_w during the training of the target model in Figures 6 and 7. Since we initiate training the controller network at Epoch 30, and we

require a mask $\mathbf{w}$ from the controller network, T_w is selected from the set 30, 50, 100, 150. In general, these values can converge to the optimal under the mask with different T_w.

Impact of Projection Operator: To validate the flexibility of our proposed algorithm to different projection operators, we plot the test accuracy using the proximal gradient projector in Equation (4-4) and the Half-Space projector in Equation (4-3). In OTO [16] and OTOv2 [17], due to the limitation of the manually selected static mask, they have to use (D)HSP [17] to achieve better performance. In our tests, we demonstrate that the controller network aids in solving these challenging issues, and there is no significant difference between the two projection operators. Due to the ease of implementation and the improved efficiency of the proximal gradient projector, we use it in our experiments.

4.4.6 Implementation Details

In this subsection, we provide more details of the implementation in the experiments about the design of the controller network and training setting.

Table 6 shows the selection of ratio p in all experiments. We choose the value of p according to the existing baseline algorithms to compare our algorithm (ATO) with others.

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 Equation (4-5) for all datasets and models is presented in Table 7.

Table 6: Choice of p in Equation (4-5) 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 [1] in OTOv2 [17] 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 [89]. As mentioned before, we set T_{start} to around 10% of the total training epochs and T_{end} to 50% of the total training epochs.

Table 7: 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 $\rightarrow \mathbf{w}$	-

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

Table 7 shows the architecture of the controller network. We make efforts to reduce extra computational overhead due to the use of controller network. Controller Network uses bi-directional gated recurrent units (GRU) [19] since it is a lightweight sequence model. It is 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. In this case, we can batch ZIGs $\mathcal{G}$.

The model mask vector is generated as below:

$$\mathbf{w} = \text{round}(\text{sigmoid}((o + s + b)/\tau)) \tag{4-50}$$

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

In the implementation, we introduce a module called the virtual gate following ZIGs $\mathcal{G}$ to automatically control the output of each ZIG in the target model based on the mask from the controller network. Since OTOv2 proposes how to automatically construct ZIGs $\mathcal{G}$ and the construction of the controller network from ZIGs is straightforward, it won't introduce much manual effort.

4.5 Summary

In this study, we explore the realm of automatic network pruning from scratch, aiming to address the shortcomings identified in existing algorithms. These limitations include 1) the involvement of intricate multi-step training procedures and 2) suboptimal outcomes associated with the static selection of pruning groups.

Our proposed solution, Auto-Train-Once (ATO), presents an innovative network pruning algorithm designed to automatically reduce the computational and storage costs of Deep Neural Networks (DNNs) without the need for an additional fine-tuning step. During the

model training phase, a controller network dynamically generates a binary mask, guiding the pruning process for the target model. With careful design of the controller network, the extra computational overhead will be less than 5%.

Furthermore, we introduce a novel stochastic gradient algorithm that provides flexibility in choosing projection operators, enhancing the coordination between model training and controller network training for improved pruning performance. We also provide a theoretical analysis for proposed algorithm, under mild assumptions, ensuring convergence, and conduct extensive experiments. The results of these experiments 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, and ImageNet.

In summary, the main contributions of this chapter 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 running, 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 OTO and OTOv2. Extensive experiments conducted on CIFAR-10, CIFAR-100, and ImageNet show that our method outperforms existing methods.

5.0 Task 3: Leveraging Foundation Models in Efficient Federated Learning

5.1 Background

In this chapter, we continue to explore the non-convex model compression. We focus on the more complicated case and put the federated learning into consideration. Since the development of the foundation mode, we utilize the model distillation in this chapter. The work in this chapter was published in the NeurIPS2023 FL workshop [102].

Foundation models exhibit remarkable effectiveness across a wide range of machine-learning tasks and applications. Many different foundation models are proposed. The CLIP (Contrastive Language–Image Pretraining) model as shown in 10 and introduced by Radford et al. [87], is specifically designed to establish an understanding and connection between images and text within a shared embedding space. CLIP excels in comprehending the relationships between images and their corresponding textual descriptions, enabling its application to various tasks involving both modalities. What sets CLIP apart is its remarkable ability to generalize across different tasks without extensive fine-tuning. By learning to associate images and text, CLIP brings similar pairs closer together in the embedding space while pushing dissimilar pairs apart. This unique property allows CLIP to be effectively utilized in diverse applications, including image classification, object detection, and text-based image retrieval, without necessitating task-specific adaptations.

Federated learning (FL) was introduced as a crucial distributed training paradigm in large-scale machine learning to mitigate communication overhead. In the FL setting, a central server orchestrates multiple worker nodes to collaboratively learn a joint model through periodic model averaging, utilizing only the local data of each worker node. This distributed approach shares the computational load among worker nodes and offers a degree of data privacy in the FL process. However, when confronted with non-iid data distributions in the context of federated training (FL), challenges related to convergence and suboptimal model performance may arise, which is called client drift, as shown in 11. Integrating foundation models into the framework of federated learning holds the potential to address issues

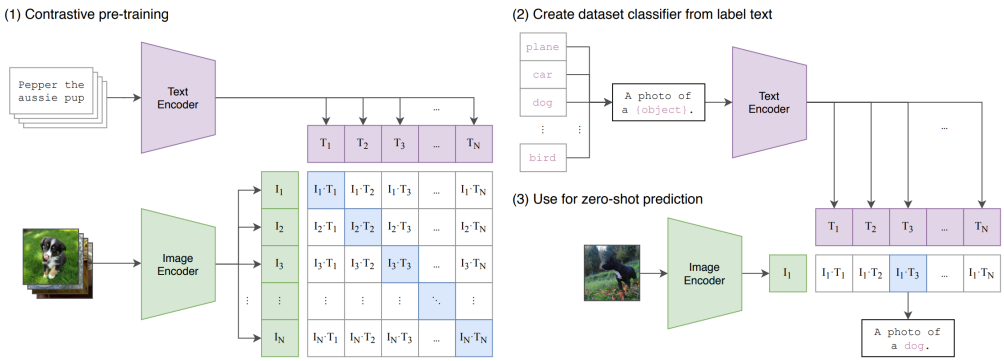


Figure 10: Summary of the CLIP model). [51].

stemming from non-iid distributions, thereby improving the overall effectiveness of the FL process.

5.2 Distilling Foundation Models in Federated Learning

5.2.1 Federate Learning: Setup

A federated learning problem could be defined as

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta) := \sum_{i=1}^{N} p_i \mathcal{L}(D_i; \theta). \tag{5-1}$$

where $\theta \in \mathbb{R}^d$ denotes the model parameter and N indicates the number of worker nodes. Each client-i has its local private dataset $\mathcal{D}_i$. The local loss function relevant to the i-th client is represented as $\mathcal{L}(D_i; \theta)$. When $\mathcal{D}_i$ and $\mathcal{D}_j$ are different ($i \neq j$), it is referred to as the heterogeneous data setting. In this dissertation, $\{\mathcal{D}_i\}_{i=1}^N$ are not identical since we consider the Non-IID distribution in the federated learning. Furthermore, p_i is a re-weighting factor conditioned as $p_i \geq 0$ and $\sum_i p_i = 1$. Usually, we assume $p_i = \frac{|\mathcal{D}_i|}{\sum_{j \in \mathcal{S}_t} |\mathcal{D}_j|}$ where S_t represents the collection of clients engaging in communication with the server at round t.

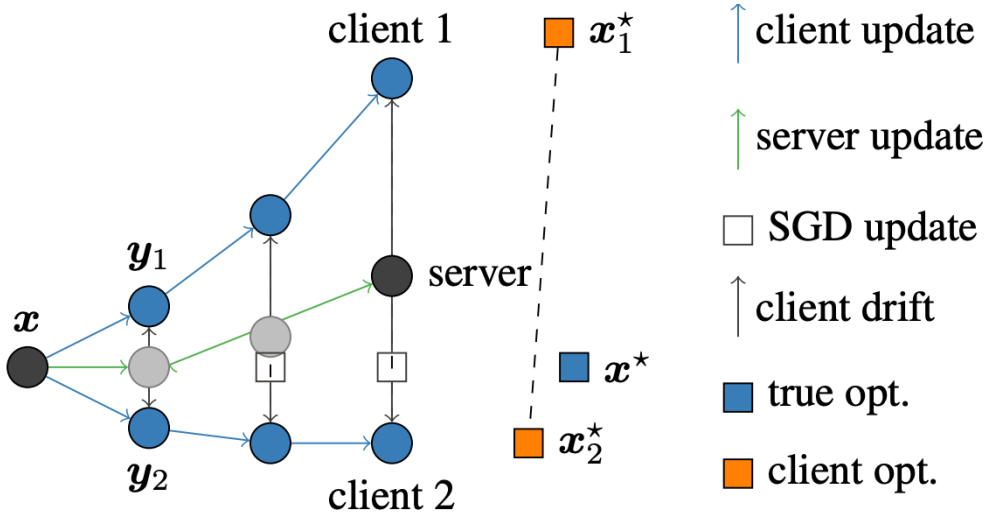


Figure 11: Client-drift in FEDAVG is illustrated for 2 clients with 3 local steps (N = 2, K = 3). The local updates y_i (in blue) move towards the individual client optima x_i^* (orange square). The server updates (in red) move towards $\frac{1}{N}\sum_i x_i^*$ instead of to the true optimum x^* (black square) [51].

FL framework repeats the following steps until the global model converges: 1) The server broadcasts the present global model to chosen clients; 2) Each client resets its local model using the received model, conducts local training based on local data, and transmits the updated weights/gradients to the server; 3) The central server adjusts the global model by aggregating the received weights/gradients.

5.2.2 Fed-LPFM

Preliminary Experiment First of all, we did a preliminary test about the application of foundation models in federated learning compared with lightweight models. We see as the increasing of data heterogeneity, the performance of lightweight models drops largely without any extra guidance. Take mobile net with CIFAR-10 as an example, it could reach 91.48% in the IID distribution. However, if we increase data heterogeneity, it drops to 90.86% in the Dirichlet ($\alpha = 1.0$), 90.05% in the Dirichlet ($\alpha = 0.5$), 84.78% in the Dirichlet ($\alpha = 0.1$), 80.48% in the Dirichlet ($\alpha = 0.05$), and 58.49% in the Dirichlet ($\alpha = 0.01$). However, the performance of CLIP model wth FedAvg is very stable and all performance are better than 90%. This preliminary test shows that foundation models have remarkable performance in the federated learning, regardless of data heterogeneity. However, there is a conflict between the large size of foundation models and the capacity of edge devices in federated learning. Therefore, we propose our method Fed-LPFM to solve this issue.

Setup Distinctive to our framework, we consider a scenario in which each client possesses only access to locally pre-trained foundation models. Much like the training dataset exclusive to each client, these foundation models are only accessible to the respective client and remain private within the Federated Learning (FL) system. We also assume that each client comprises two sets of local models: (a) a set M_i of pre-trained foundation models (private), denoted as $\mathcal{M}_i^1, \mathcal{M}_i^2, \ldots, \mathcal{M}_i^{M_i}$, and (b) a trainable small-scale lightweight model parameterized by θ_i. Given the privacy of the foundation models, only the lightweight models circulate among the clients and the server, facilitating the exchange of knowledge across the entire system. Our objective is to minimize the function in Equation (5-1), where the optimization of θ corresponds to the parameters of the small-scale lightweight model, leaving

the foundation models unaltered.

Local Training Within our algorithm, the client leverages its locally stored data in conjunction with insights from its confidential foundation models to oversee local training. To achieve this, we employ the following loss function:

$$\mathcal{L}(D_i;\theta) = \lambda\mathcal{L}_{CE}(D_i;\theta) + (1-\lambda)\mathcal{L}_{Distill}(D_i;\theta,\mathcal{M}_i^1,\ldots,\mathcal{M}_i^{M_i}). \tag{5-2}$$

Here, the first term is the local cross-entropy loss, denoted as

$$\mathcal{L}_{CE}(D_i;\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\ell_{CE}(h(x;\theta),y), \tag{5-3}$$

where $h(\cdot)$ denotes the outcome of a forward pass through the lightweight model. The second term, $\mathcal{L}_{Distill}$, serves the purpose of distilling knowledge between the lightweight model and the pre-trained foundation models. Traditionally, the Kullback Leibler (KL) Divergence loss is employed for this task.

$$\mathcal{L}_{Distill}(D_i;\theta,\mathcal{M}_i^1,\ldots,\mathcal{M}_i^{M_i}) = \sum_{m=1}^{M_i}\mathbb{E}_{(x,y)\sim\mathcal{D}_i}\ell_{KL}[h(x;\theta)||\mathcal{M}_i^m(x)]. \tag{5-4}$$

The parameter λ controls the proportion of knowledge distilled from the foundation model in comparison to ground-truth labels.

Aggregation scheme Following local training, the server syncs up with the accessible clients and consolidates the locally updated lightweight models. The aggregation of the local models employs the subsequent re-weighting scheme:

$$\theta_{t+1} = \sum_{i\in S_t}\frac{|D_i|}{\sum_{j\in S_t}|\mathcal{D}_j|}\theta_t^i, \tag{5-5}$$

where t is the communication round. Once the aggregation is finalized, the server disseminates the updated model to clients, and the entire process iterates until a specified termination condition is achieved. The details are shown in Algorithm 10.

Algorithm 10 Fed-LPFM

Input: Dataset $\mathcal{D}_i$, frozen and private pre-trained foundation models: $\mathcal{M}_i^1, \mathcal{M}_i^2, \ldots, \mathcal{M}_i^{M_i}$ and lightweight model θ_0 for each client $i \in [N]$.

Server:

for Round $t = 0, 1, 2, \ldots, T-1$ **do**

  Send θ_t to connected clients $\mathcal{S}_t \subset [N]$. Let $P_t = \sum_{i \in \mathcal{S}_t} |\mathcal{D}_i|$.

  for Client $i \in \mathcal{S}_t$ in parallel **do**

    $\theta_t^i \leftarrow$ **LocalUpdate**(θ_t, i)

    Send the updated model θ_t^i to the central server

  end for

  Server-end aggregation: $\theta_{t+1} = \sum_{i \in \mathcal{S}_t} \frac{|D^i|}{P_t} \theta_t^i$

end for

return: θ_T

LocalUpdate(θ_t, i)

$\theta_t^i = \theta_t$

for epoch $q = 0, 1, \ldots, Q-1$: $\theta_{t,q+1}^i = \theta_{t,q}^i - \eta \tilde{\nabla} \mathcal{L}(\theta_{t,q}^i; \mathcal{M}_1^i, \ldots, \mathcal{M}_{M_i}^i; \mathcal{D}_i)$

return: $\theta_t^i = \theta_{t,Q}^i$

5.3 Experiments

5.3.1 Experiments Setup

Data Settings We assess the performance of our algorithm on the CIFAR-10 dataset involving 10 clients distributed across seven data partitions, characterized by varying levels of heterogeneity, including both IID and non-IID scenarios. For the non-IID data partitions, we employ (1) the Dirichlet distribution, denoted as Dir(α) with $\alpha = 1.0,\ 0.5,\ 0.1,\ 0.05,\ 0.01$; (2) Class Split, where each client's data is sampled from 2 of the 10 classes. The evaluation

encompasses all algorithms on the balanced CIFAR-10 test set, and we present the average accuracy over three trials to mitigate the impact of random variability in the runs.

Network Architectures For the choice of foundation models, we utilize CLIP [87] with ViT-Base/32 (default) and RN50 as backbones and employ MobileNet-v2, EfficientNetB0, and ResNet18 as our lightweight models. In the case of each lightweight model, we substitute batch normalization layers with group normalization (8 groups) and initiate training from random initialization.

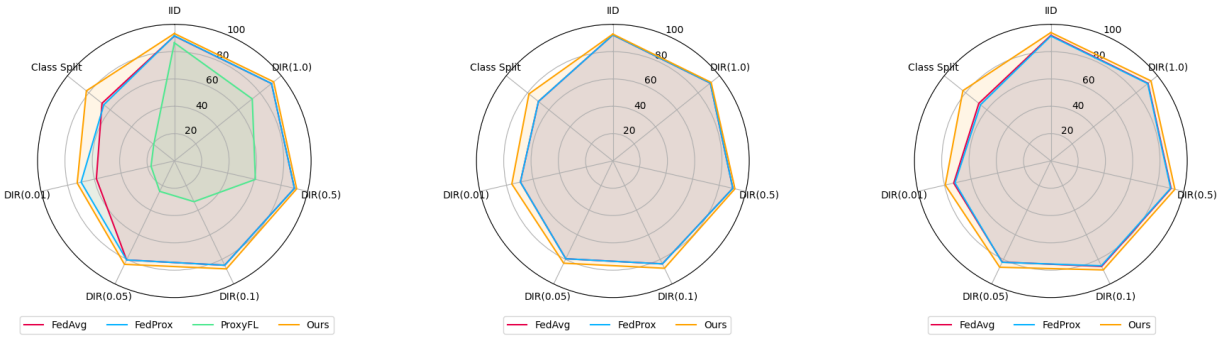
Training Setup and Hyper-parameters Across all algorithms and experiments, we employ an SGD optimizer for training. The lightweight models undergo 600 epochs of training with a learning rate of 0.01, weight decay of 5e-4, and a step learning rate scheduler incorporating a scale factor of 0.1 at epoch 200. In ablation studies, where we also explore direct fine-tuning of foundation models, the training spans 200 epochs with a learning rate of 2e-3, weight decay of 5e-4, and a cosine learning rate scheduler with a warmup period of 1 epoch. When comparing against state-of-the-art (SOTA) algorithms, we train the lightweight models for up to 500 epochs in FedAvg and FedProx, and 600 epochs in FML.

5.3.2 Empirical Results

The tabulated experimental results, utilized to generate figures in the main dissertation, demonstrate the consistent superiority of our algorithm, Fed-LPFM, over other algorithms. Notably, Fed-LPFM exhibits the most substantial improvement in scenarios involving class split and Dirichlet distribution ($\alpha = 0.01$ and 0.05), which represent highly heterogeneous data distributions across various clients. For instance, in the class split scenario, Fed-LPFM achieves a 21.6% increase compared to Fedavg (82.29% vs. 67.67%).

Similar results are evident across various proxy model backbones. In both the cases of EfficientNet and ResNet, Fed-LPFM consistently surpasses other methods across all data heterogeneity settings.

As supporting materials for Figure 14 (a), we report the numerical results for testing fine-tuned CLIP (linear probing and prompt tuning) and zero-shot CLIP cross different data heterogeneity levels. We observed that zero-shot CLIP offers best and more robust



(a) MobileNetV2 (b) EfficientNetB0 (c) ResNet18

Figure 12: When assessing performance against existing works across diverse data settings and various lightweight model backbones, Fed-LPFM consistently surpasses previous approaches by a significant margin, particularly in scenarios with highly heterogeneous data distributions. A broader coverage area indicates a more robust and effective federated learning approach.

Table 8: The values of Figure 12 (a) are shown in the current table.

Data Settings	FedAvg	FedProx	FML	Fed-LPFM (MobileNetV2)
Class Split	67.67 ± 1.88	65.97 ± 5.07	19.29 ± 0.28	$\mathbf{82.29 \pm 1.12}$
Dir(0.01)	58.49 ± 15.72	69.92 ± 2.07	17.50 ± 1.22	$\mathbf{72.88 \pm 9.11}$
Dir(0.05)	80.48 ± 1.07	80.48 ± 1.07	24.75 ± 3.15	$\mathbf{84.05 \pm 0.99}$
Dir(0.1)	84.78 ± 1.65	84.78 ± 1.65	33.13 ± 2.58	$\mathbf{87.73 \pm 1.48}$
Dir(0.5)	90.05 ± 0.21	90.05 ± 0.21	60.77 ± 3.98	$\mathbf{91.72 \pm 0.31}$
Dir(1.0)	90.86 ± 0.13	90.86 ± 0.13	72.84 ± 1.56	$\mathbf{92.87 \pm 0.17}$
IID	91.48 ± 0.31	91.61 ± 0.34	86.49 ± 0.17	$\mathbf{93.26 \pm 0.17}$

performances when compared to fine-tuned methods.

As supporting materials for Figure 14 (b), we report the results of using CLIP: ResNet50 and CLIP:ViT-B/32 as well as random sampling of them, with uniform prior, as foundation

Table 9: The values of Figure 12 (b) are shown in the current table.

Data Settings	FedAvg	FedProx	Fed-LPFM (EfficientNetB0)
Class Split	69.81 ± 2.64	69.81 ± 2.64	$\mathbf{78.70 \pm 2.28}$
Dir(0.01)	69.56 ± 2.20	69.56 ± 2.20	$\mathbf{75.98 \pm 1.29}$
Dir(0.05)	79.53 ± 0.95	79.53 ± 0.95	$\mathbf{83.12 \pm 0.76}$
Dir(0.1)	83.67 ± 1.23	83.67 ± 1.23	$\mathbf{87.24 \pm 1.46}$
Dir(0.5)	90.21 ± 0.08	90.21 ± 0.08	$\mathbf{91.80 \pm 0.30}$
Dir(1.0)	91.41 ± 0.35	91.41 ± 0.35	$\mathbf{92.17 \pm 0.09}$
IID	92.22 ± 0.17	92.22 ± 0.17	$\mathbf{92.84 \pm 0.04}$

Table 10: The values of Figure 12 (c) are shown in the current table.

Data Settings	FedAvg	FedProx	Fed-LPFM(ResNet18)
Class Split	67.56 ± 4.66	65.99 ± 6.41	$\mathbf{82.50 \pm 2.00}$
Dir(0.01)	73.06 ± 2.10	72.13 ± 1.38	$\mathbf{79.41 \pm 0.95}$
Dir(0.05)	82.22 ± 0.79	82.40 ± 0.46	$\mathbf{86.42 \pm 1.33}$
Dir(0.1)	85.74 ± 1.38	85.23 ± 0.89	$\mathbf{88.59 \pm 1.54}$
Dir(0.5)	90.41 ± 0.30	90.095 ± 0.54	$\mathbf{93.30 \pm 0.10}$
Dir(1.0)	91.13 ± 0.13	90.82 ± 0.25	$\mathbf{93.92 \pm 0.14}$
IID	91.94 ± 0.22	91.45 ± 0.17	$\mathbf{94.00 \pm 0.20}$

models. It shows that random selection of pre-trained models offers worst performances when compared to the other two.

Table 11: The values of Figure 14(a) are shown in the current table.

Data Settings	Linear Probing	Prompt Tuning	0-shot (Ours)
Class Split	45.63 ± 9.48	51.85 ± 4.58	$\mathbf{82.29 \pm 1.12}$
Dir(0.01)	44.76 ± 3.56	35.86 ± 6.03	$\mathbf{72.88 \pm 9.11}$
Dir(0.05)	73.07 ± 0.86	68.39 ± 1.65	$\mathbf{84.05 \pm 0.99}$
Dir(0.1)	82.54 ± 3.21	77.35 ± 3.75	$\mathbf{87.73 \pm 1.48}$
Dir(0.5)	91.08 ± 0.24	89.01 ± 0.12	$\mathbf{91.72 \pm 0.31}$
Dir(1.0)	92.53 ± 0.48	90.32 ± 0.33	$\mathbf{92.87 \pm 0.17}$
IID	92.56 ± 0.17	91.85 ± 0.23	$\mathbf{93.26 \pm 0.17}$

Table 12: The values of Figure 14 (b) are shown in the current table.

Data Settings	CLIP: RN50	CLIP: ViT-B/32	Random Selection
Class Split	80.46 ± 4.08	$\mathbf{82.29 \pm 1.12}$	62.75 ± 3.24
Dir(0.01)	66.17 ± 1.19	$\mathbf{72.88 \pm 9.11}$	64.02 ± 3.52
Dir(0.05)	$\mathbf{84.42 \pm 0.41}$	84.05 ± 0.99	71.33 ± 1.52
Dir(0.1)	$\mathbf{87.90 \pm 1.58}$	87.73 ± 1.48	76.05 ± 3.53
Dir(0.5)	$\mathbf{92.04 \pm 0.25}$	91.72 ± 0.31	84.82 ± 0.45
Dir(1.0)	$\mathbf{93.00 \pm 0.10}$	92.87 ± 0.17	83.99 ± 0.99
IID	$\mathbf{93.63 \pm 0.30}$	93.26 ± 0.17	85.01 ± 0.47

5.3.3 Results Analysis

SOTA Algorithm Comparison We compare our approach against FedAvg [80], FedProx [65], and FML [92], in various data heterogeneity settings. We show comparison results in Figure 12, where each data partition is represented as a vertex on the polar plot, and we plot accuracy along the radius. Based on the findings presented in Figure 12, it is evident

that Fed-LPFM consistently outperforms previous approaches across diverse data distributions. Notably, our algorithm demonstrates a substantial improvement over FedProx, the best-performing method among prior works, particularly in scenarios with the most extreme heterogeneous distributions, such as class split and Dirichlet sampling with $\alpha = 0.01, 0.05$.

Furthermore, it's worth noting that employing MobileNet as the backbone for both the private and lightweight models, akin to the setup in FML, yields unsatisfactory results. We hypothesize that fine-tuning on local data introduces biases into the representations learned across both models, leading to a significant deterioration in performance as data heterogeneity increases.

lightweight Model To demonstrate the versatility of our approach across different lightweight model backbones, we conduct evaluations on EfficientNetB0, ResNet18, and MobileNetV2. The results are both reported and visualized in Figure 12. It is evident from our observations that our approach consistently outperforms FedAvg and FedProx across the entire array of lightweight models, particularly in scenarios involving various levels of data heterogeneity, including severe non-IID cases. Furthermore, we note that the performance improvement achieved by FedProx diminishes across both ResNet and EfficientNet, as compared to MobileNet. FedAvg and FedProx exhibit similar performance levels in this comparison.

Fine-Tuned vs. 0-shot Our Fed-LPFM technique employs pre-trained foundation models without the option for fine-tuning. To investigate the influence of prior knowledge on distillation, we contrast our 0-shot approach with the initial step of fine-tuning each client's foundation model(s) only on **local data** using linear probing and prompt tuning as shown in Figure 13. Figure 14 (a) visually depicts the superior performance of the 0-shot CLIP case over the fine-tuned CLIP models. We posit that fine-tuning the foundation model on local data leads to a more personalized and biased knowledge representation, resulting in decreased performance on a balanced test set. Moreover, in settings with heterogeneous data distribution, locally encoded knowledge tends to be more personalized and biased. Conversely, in more homogeneous settings, there is a notable performance boost for clients leveraging knowledge from both 0-shot and fine-tuned foundation models. We attribute the improvement observed when distilling from 0-shot models to the strong diversity in feature

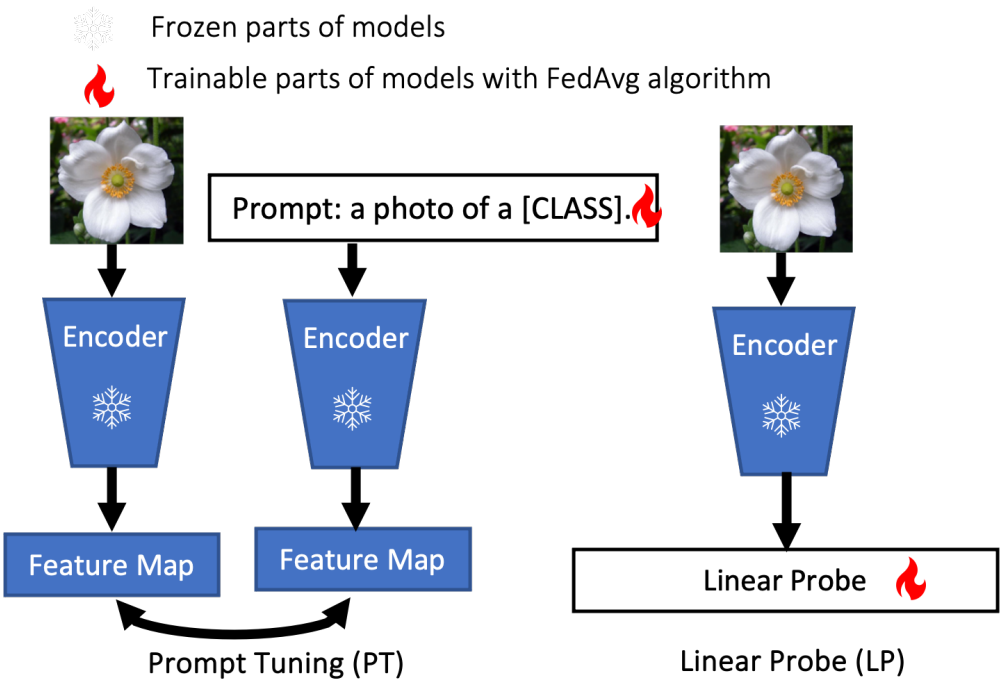
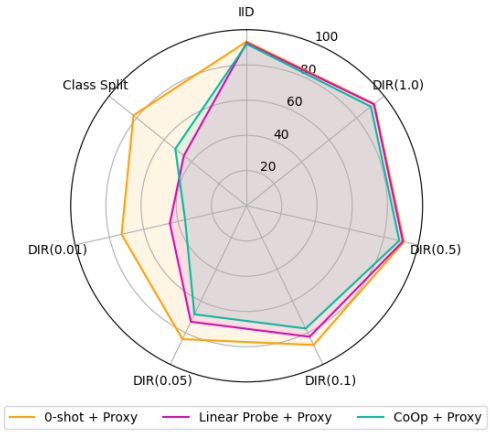
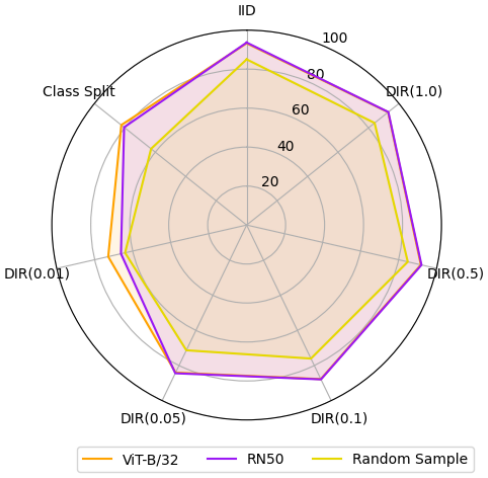


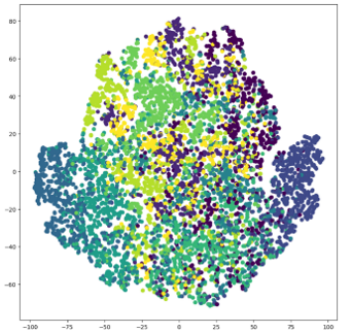
Figure 13: Overview of Fine-tuning of the CLIP model.



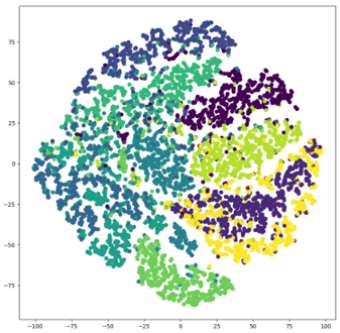
(a) Fine-tuning vs. 0-shot



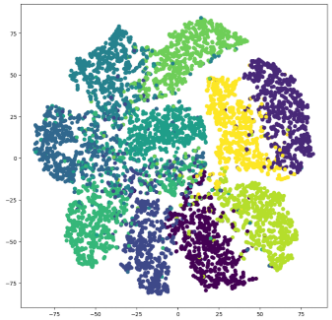
(b) Personalizing 0-shot Models



(c) Linear Probing



(d) Prompt Tuning



(e) 0-shot

Figure 14: (a) Fine-tuning foundation models on local data forces significantly worse performances under non-IID conditions. (b) Maintaining consistent foundation model backbones improves the synergy in information shared across clients. (**Bottom**) When compared to fine-tuned models, 0-shot models offer more diverse feature embeddings that reduce the bias of lightweight models towards local data distributions.

embeddings compared to fine-tuned foundation models. We utilize tSNE plots to examine the distribution of encoded knowledge representations from foundation models. As shown in Figure 14 (c) - (f), features from 0-shot foundation models exhibit a broader coverage compared to fine-tuned models.

Personalizing Foundation Models Fed-LPFM ensures the privacy of foundation model(s) by allowing each client to tailor them to their specific needs. As illustrated in Figure 14 (b), maintaining consistent backbones across the foundation models results in the most significant performance improvement, while introducing a random selection of backbones, ranging from ViT-B/32 to RN50, leads to a performance decline. We attribute this behavior to the adoption of a simplistic strategy in aggregating information from multiple lightweight models, driven by variations in the knowledge and understanding of foundation models with different backbones.

Alternatively, incorporating our approach in conjunction with personalized Federated Learning (FL) methods ([95, 26, 63, 34, 20], etc.) could potentially enhance overall performance.

5.4 Summary

In summary, we verify the outstand performance of foundation models in federated learning and proposed Fed-LPFM as an approach to harness foundation models under the interference efficiency, enhancing the performance and robustness attainable in small-scale models within the Federated Learning (FL) framework. Distilling knowledge from pre-trained foundation models, rather than fine-tuned ones, introduces the necessary diversity in feature representations to mitigate bias towards local distributions, thereby enhancing client performance across diverse data distributions. The implementation of logit-level distillation provides clients with the flexibility to select local foundation models based on individual constraints. This work points towards a crucial future direction: the exploration of an approach that synergistically combines information from diverse knowledge representations to achieve improved model performance. Overall, our main contributions are as follows:

- This marks the initial attempt to utilize foundation models in federated learning through distillation, aiming to enhance the performance and robustness attainable in small-scale client models (e.g., a notable increase of 9.22% for EfficientNetB0, 8.69% for ResNet18, and 24.60% for MobileNetV2).

- In the realm of low-latency models, we present a federated learning solution that remains resilient in the face of diverse and heterogeneous client data. Our approach surpasses previous methodologies across a spectrum of client data distributions, ranging from IID to various parametrized Dirichlet distributions and class-specific partitions.
- We investigate the influence of utilizing representations from fine-tuned foundation models on local data compared to pre-trained foundation models. Our findings reveal that under IID data distributions, the initial fine-tuning of foundation models provides no advantage over 0-shot foundation models. Moreover, as data heterogeneity increases, accuracy is significantly impeded, implying that direct fine-tuning of foundation models results in biased representations.
- Our framework introduces a novel feature by granting clients the flexibility to choose their locally-stored foundation models (personalization) based on the available compute and data scale. We analyze the influence of diverse foundation model backbones and underscore the significance of appropriately combining disparate feature representations.

6.0 Conclusion

In this dissertation, we study the application of convex and non-convex model compression, including model screening, structural pruning, and model distillation, in ML training. We propose corresponding algorithms for convex models and non-convex models and consider centralized training or multi-party training. With the model compression technique, we could reduce communication overhead, speed up model training, and improve inference efficiency. In addition, extensive experiments and related theoretical analysis are provided. Overall, our work can be summarized as below:

- In the first chapter, we consider convex models in the distributed setting and we introduce a novel asynchronous learning approach (DDSS) tailored for stochastic composite optimization. To the extent of my understanding, this marks the first effort in the realm of distributed dynamic safe screening. Theoretical analysis is provided and shows that our proposed approach attains a linear convergence rate while maintaining lower overall complexity.

  Additionally, our method effectively eliminates nearly all inactive variables within a finite number of iterations with high probability. Empirical findings substantiate the effectiveness and superiority of our approach since it dynamically removes the abundant feature in advance and speeds up the training largely. The test results also show it has the linear speed-up property and training time decreases as the increase of number of workers. Notably, our focus here is on the block-separable norm; exploring the extension of our method to non-separable norms, such as OWL regression represents a promising avenue for future research.
- Afterwards, we focus on the sparse representation of deep (non-convex) models. This paper delves into the realm of automatic network pruning from scratch in centralized learning. It addresses shortcomings identified in existing algorithms, notably: 1) intricate multi-step training procedures, and 2) suboptimal outcomes associated with statically chosen pruning groups.

  Our proposed algorithm (ATO) introduces an innovative network pruning algorithm ex-

plicitly crafted to automatically reduce the computational and storage costs of Deep Neural Networks (DNNs) without necessitating an additional fine-tuning step. In the model training phase, a controller network dynamically generates a binary mask, guiding the pruning process for the target model. Furthermore, we have devised a novel stochastic gradient algorithm, providing flexibility in the selection of projection operators and enhancing coordination between model training and controller network training, thereby improving pruning performance. The results of these experiments showcase that our algorithm attains 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.

Additionally, we present a general theoretical analysis with mild assumptions to ensure convergence. The theoretical analysis uses the framework of mirror descent and it consists of various optimizers, such as non-adaptive optimizers (SGD) and adaptive optimizer (ADAM).

- Finally, we delve into the utilization of foundational models within the context of federated learning. Federated learning is similar to distributed training but it is more flexible since it only selects a subset of clients in each round and the client can perform multiple local training to save communication overhead.

  Our investigation highlights the impressive performance achieved by foundation models following fine-tuning, particularly in scenarios characterized by heterogeneous distribution. Furthermore, we illustrate how the incorporation of a foundation model can assist smaller models in mitigating the effects of client drift to a certain degree.

  We introduce Fed-LPFM as a strategy to harness foundation models, enhancing the performance and robustness attainable in small-scale models within the Federated Learning (FL) framework. Distillation from pre-trained foundation models, rather than fine-tuned ones, introduces diversity in feature representations, mitigating bias towards local distributions and thereby enhancing client performance across various heterogeneous data distributions. The implementation of logit-level distillation affords clients the flexibility to select local foundation models based on their individual constraints. This work establishes a crucial avenue for future research: the exploration of an approach that synergistically

combines information from diverse knowledge representations to enhance overall model performance.

In conclusion, this study has systematically study the convex and non-convex model compression. It also consider both centralized learning and multi-part collaborative training. But the experiments only consider the task in the classification as well as computing vision task. Future work could consider a wider range of ML tasks, such as natural language processing or object detection.

Bibliography

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2020.

[2] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International conference on machine learning*, pages 699–707. PMLR, 2016.

[3] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

[4] Runxue Bao, Bin Gu, and Heng Huang. Efficient approximate solution path algorithm for ordered weighted l_1-norm with accuracy guarantee. In *IEEE ICDM*, 2019.

[5] 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.

[6] Runxue Bao, Xidong Wu, Wenhan Xian, and Heng Huang. Doubly sparse asynchronous learning for stochastic composite optimization. In *IJCAI*, pages 1916–1922, 2022.

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

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

[9] Wanyu Bian, Yunmei Chen, Xiaojing Ye, et al. An optimization-based meta-learning model for mri reconstruction with diverse dataset. *J. Imaging*, 2021.

[10] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[11] Yair Censor and Arnold Lent. An iterative row-action method for interval convex programming. *Journal of Optimization theory and Applications*, 34(3):321–353, 1981.

[12] Yair Censor and Stavros Andrea Zenios. Proximal minimization algorithm withd-functions. *Journal of Optimization Theory and Applications*, 73(3):451–464, 1992.

[13] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM TIST*, 2011.

[14] Jinyu Chen, Wenchao Xu, Song Guo, Junxiao Wang, Jie Zhang, and Haozhao Wang. Fedtune: A deep dive into efficient federated fine-tuning with pre-trained transformers. *arXiv preprint arXiv:2211.08025*, 2022.

[15] 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.

[16] 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.

[17] Tianyi Chen, Luming Liang, Tianyu Ding, Zhihui Zhu, and Ilya Zharkov. Otov2: Automatic, generic, user-friendly. *arXiv preprint arXiv:2303.06862*, 2023.

[18] Yunmei Chen, Hongcheng Liu, Xiaojing Ye, et al. Learnable descent algorithm for nonsmooth nonconvex image reconstruction. *SIAM J. Imaging Sci.*, 2021.

[19] 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.

[20] Yae Jee Cho, Jianyu Wang, Tarun Chiruvolu, and Gauri Joshi. Personalized federated learning for heterogeneous clients with clustered knowledge transfer. *arXiv preprint arXiv:2109.08119*, 2021.

[21] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019.

[22] Rudrajit Das, Anish Acharya, Abolfazl Hashemi, Sujay Sanghavi, Inderjit S Dhillon, and Ufuk Topcu. Faster non-convex federated learning via global and local momentum. In *Uncertainty in Artificial Intelligence*, pages 496–506. PMLR, 2022.

[23] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27:1646–1654, 2014.

[24] Aaron Defazio and Léon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. *arXiv preprint arXiv:1812.04529*, 2018.

[25] 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.

[26] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33, 2020.

[27] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in Neural Information Processing Systems*, 31, 2018.

[28] 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.

[29] Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Mind the duality gap: safer rules for the lasso. In *ICML*, 2015.

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

[31] 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.

[32] 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.

[33] Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination in sparse supervised learning. Technical report, 2010.

[34] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[35] Bin Gu, Zhouyuan Huo, and Heng Huang. Asynchronous doubly stochastic group regularized learning. In *AISTATS*, 2018.

[36] 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.

[37] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[38] 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.

[39] 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.

[40] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[41] 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.

[42] 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.

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

[44] 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.

[45] Zhouyuan Huo and Heng Huang. Asynchronous mini-batch gradient descent with variance reduction for non-convex optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[46] 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*, ICML, pages 448–456. JMLR.org, 2015.

[47] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

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

[49] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

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

[51] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[52] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

[53] Prashant Khanduri, Pranay Sharma, Haibo Yang, Mingyi Hong, Jia Liu, Ketan Rajawat, and Pramod Varshney. Stem: A stochastic two-sided momentum algorithm achieving near-optimal sample and communication complexities for federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[54] 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.

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

[56] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

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

[58] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Asaga: asynchronous parallel saga. In *AISTATS*, 2017.

[59] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Improved asynchronous parallel optimization analysis for stochastic incremental methods. *JMLR*, 2018.

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

[61] Junyi Li, Bin Gu, and Heng Huang. A fully single loop algorithm for bilevel optimization without hessian inverse. *arXiv preprint arXiv:2112.04660*, 2021.

[62] Qingyang Li, Shuang Qiu, Shuiwang Ji, et al. Parallel lasso screening for big data optimization. In *ACM SIGKDD*, 2016.

[63] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

[64] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[65] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[66] 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.

[67] 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.

[68] 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.

[69] Yitan Li, Linli Xu, Xiaowei Zhong, and Qing Ling. Make workers work harder: decoupled asynchronous proximal stochastic gradient descent. *arXiv preprint arXiv:1605.06619*, 2016.

[70] 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.

[71] 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.

[72] Xianfeng Liang, Shuheng Shen, Jingchang Liu, Zhen Pan, Enhong Chen, and Yifei Cheng. Variance reduced local sgd with lower communication complexity. *arXiv preprint arXiv:1912.12844*, 2019.

[73] 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.

[74] 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.

[75] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[76] 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.

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

[78] Wang Lu, Xixu Hu, Jindong Wang, and Xing Xie. Fedclip: Fast generalization and personalization for clip in federated learning. *arXiv preprint arXiv:2302.13485*, 2023.

[79] Michael Lustig, David L Donoho, Juan M Santos, and John M Pauly. Compressed sensing mri. *IEEE Signal Process. Mag.*, 2008.

[80] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[81] Qi Meng, Wei Chen, Jingcheng Yu, et al. Asynchronous stochastic proximal optimization algorithms with variance reduction. In *AAAI*, 2017.

[82] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019.

[83] Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, et al. Gap safe screening rules for sparsity enforcing penalties. *JMLR*, 2017.

[84] Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Gap safe screening rules for sparse-group lasso. In *NeurIPS*, 2016.

[85] Fabian Pedregosa, Rémi Leblond, and Simon Lacoste-Julien. Breaking the nonsmooth barrier: A scalable parallel method for composite optimization. In *NeurIPS*, 2017.

[86] 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.

[87] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[88] Alain Rakotomamonjy, Gilles Gasso, and Joseph Salmon. Screening rules for lasso with non-convex sparse regularizers. In *ICML*, 2019.

[89] 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.

[90] Felix Sattler, Tim Korjakow, Roman Rischke, and Wojciech Samek. Fedaux: Leveraging unlabeled auxiliary data in federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[91] Hyowoon Seo, Jihong Park, Seungeun Oh, Mehdi Bennis, and Seong-Lyun Kim. 16 federated knowledge distillation. *Machine Learning and Wireless Communications*, page 457, 2022.

[92] Tao Shen, Jie Zhang, Xinkang Jia, Fengda Zhang, Gang Huang, Pan Zhou, Fei Wu, and Chao Wu. Federated mutual learning. *CoRR*, abs/2006.16765, 2020.

[93] Atsushi Shibagaki, Masayuki Karasuyama, Kohei Hatano, and Ichiro Takeuchi. Simultaneous safe screening of features and samples in doubly sparse modeling. In *ICML*, 2016.

[94] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[95] Canh T Dinh, Nguyen Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33, 2020.

[96] 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.

[97] Robert Tibshirani, Jacob Bien, Jerome Friedman, et al. Strong rules for discarding predictors in lasso-type problems. *JRSS*, 2012.

[98] 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.

[99] Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? In *International Conference on Machine Learning*, pages 10334–10343. PMLR, 2020.

[100] Xidong Wu, Shangqian Gao, Zeyu Zhang, Zhenzhen Li, Runxue Bao, Yanfu Zhang, Xiaoqian Wang, and Heng Huang. Auto-train-once: Controller network guided automatic network pruning from scratch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[101] Xidong Wu, Feihu Huang, Zhengmian Hu, and Heng Huang. Faster adaptive federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10379–10387, 2023.

[102] 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.

[103] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIOPT*, 2014.

[104] An Xu and Heng Huang. Coordinating momenta for cross-silo federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8735–8743, 2022.

[105] Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-iid federated learning. *arXiv preprint arXiv:2101.11203*, 2021.

[106] 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.

[107] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

[108] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pages 7184–7193. PMLR, 2019.

[109] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700, 2019.

[110] 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.

[111] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *JRSS*, 2006.

[112] Bai Zhang, Yi Fu, Yingzhou Lu, et al. Ddn2.0: R and python packages for differential dependency network analysis of biological systems. *bioRxiv*, 2021.

[113] Haoran Zhang, Zhenzhen Hu, Wei Qin, Mingliang Xu, and Meng Wang. Adversarial co-distillation learning for image recognition. *Pattern Recognition*, 111:107659, 2021.

[114] Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. *arXiv preprint arXiv:2005.11418*, 2020.

[115] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328, 2018.

[116] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[117] Xingyu Zhou. On the fenchel duality between strong convexity and lipschitz continuous gradient. *arXiv preprint arXiv:1803.06573*, 2018.

[118] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pages 12878–12889. PMLR, 2021.

[119] 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.

[120] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex Smola. Parallelized stochastic gradient descent. *Advances in neural information processing systems*, 23, 2010.