Collaborative Distributed Machine Learning: From Knowledge Reuse to Sparsification in Federated Learning

Qianyu Long

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow

July 2024

Abstract

Distributed Machine Learning (DML) leverages distributed computing resources to train models and perform inference on decentralized datasets efficiently. A high-quality distributed system ensures optimal Quality of Service (QoS) by delivering low latency, high reliability, efficient resource utilization, and robust security. However, DML frameworks face significant challenges with the proliferation of devices generating vast volumes of data and the increasing complexity of tasks. For instance, heterogeneous feature spaces from data generated by different users or locations can undermine model reliability. Additionally, the growing size and complexity of models impose substantial burdens on resource-constrained devices, particularly for inference and storage. Latency becomes a critical concern in distributed online systems such as intelligent transport systems for autonomous vehicles.

This work explores leveraging knowledge reuse, a key meta-learning technique, combined with sparsification methods to build efficient and effective distributed learning systems. To enhance efficiency, we aim to reduce redundant computation and communication, assuming that distributed data exhibit similarities despite not being identical. Specifically, we propose identifying reusable models by examining statistical patterns and meta-features derived from trained models. These reusable models are selected and adapted to local environments without requiring full retraining. Multi-task learning further improves the effectiveness of these adaptations, ensuring comparable performance to locally trained models while significantly reducing the number of models that need to be trained. By clustering models with shared characteristics, the system reduces the computational and communication overhead in a network of M devices, where only $K \ll M$ models need to be trained, maintaining strong overall performance.

To tackle real-world challenges where data is often non-independent and identically distributed (non-i.i.d.), we incorporate pruning techniques to enhance both system efficiency and effectiveness. This approach reduces communication and computation costs while simultaneously improving model performance and accuracy. Centralized federated learning (CFL) relies on a central server for model aggregation, while decentralized federated learning (DFL) operates without central server coordination, enabling direct communication between clients. In CFL, dynamic pruning strategies with error feedback and adaptive

regularization achieve extremely sparse models, reducing computation and communication costs while accelerating inference. These models retain high sparsity with minimal accuracy loss. In DFL, the absence of a central node enhances robustness against adversarial attacks. Efficiency is further improved through dynamic pruning, allowing progressively sparser training, and a hybrid approach combining sequential and parallel training to reuse updates within the same round. Personalized pruning masks address data heterogeneity across clients, promoting both system efficiency and local model performance.

The proposed framework is validated experimentally across diverse datasets and models, including air pollution data from weather stations, temperature data from unmanned surface vehicles, and image classification tasks. The tested models span traditional approaches such as regression and support vector machines to modern deep learning architectures like convolutional neural networks. Theoretically, we conduct hypothesis testing and complexity analysis, including the development of convergence theorems for federated learning scenarios.

Overall, this thesis presents comprehensive frameworks and algorithms backed by robust experimental and theoretical results. It addresses key challenges in federated learning and edge computing through knowledge reuse and sparsification, enhancing the efficiency, effectiveness, and robustness of modern AI applications.

Contents

List of Tables

List of Figures

Acknowledgements

It has been almost ten years since I began my journey in higher education, spanning diverse locations from Macau, SAR, China, to Canberra, Australia, and Glasgow, United Kingdom. During this time, my academic focus evolved from Math to Statistics and ultimately to Computer Science, allowing me to apply theoretical knowledge to practical problems. This journey was not without challenges, but overcoming these setbacks has led me to become a researcher in Computing Science. I am deeply grateful to everyone who supported me in achieving these milestones.

First, I would like to express my sincere gratitude to my PhD supervisors, Dr. Christos Anagnostopoulos and Dr. Fani Deligianni, for their patience and exceptional expertise. Their support extended beyond research, as they also showed care for my personal well-being, making my PhD journey both happy and enriching. I am also grateful to Dr. Yanrong Yang and Dr. Daning Bi, who introduced me to academic research during my Master's studies and provided valuable career guidance. Additionally, I would like to thank Professor Jianwei Zhang for giving me the opportunity to explore Computer Science through an internship in his group, which ignited my passion for this field.

Secondly, I would also like to thank my colleagues and friends at the University of Glasgow for their collaboration and company. Specifically, I am grateful for the valuable suggestions from Dr. Sham Puthiya Parambath and Saleh Abdullah M AlFahad in our KDES group. The routine dinner gatherings have improved my cooking skills and enhanced our friendships. I want to extend my thanks to Dr. Siwei Liu, Kai Feng, Zhaohan Meng, Cong Fu, Xicheng Li, Qiyuan Wang, Zhuoran Tan, Hong Lin, Xinyu Li, and many others.

The most important is to express my deep thanks to my loved family members, including my parents, Professor Hongyan Wang and Weihong Long, and my grandparents, Shunying Li, Dexin Wang, Juzhen Yang, and Desheng Long, who have supported me unconditionally and brought me into this world. I am also deeply grateful to my girlfriend and soul mate, Xuexue Zhao, for her patience and love during this challenging period.

To conclude, I would like to share a Chinese poem that reflects my journey:

肩鸿任钜踏歌行，功不唐捐玉汝成。

Declaration

With the exception of chapters 1 and 2, which contain introductory material, all work in this thesis was carried out by the author unless otherwise explicitly stated.

Chapter 1

Introduction

1.1 Motivation

Machine learning is pivotal in the Artificial Intelligence (AI) era. Statistical machine learning models, such as Support Vector Machines (SVM), Decision Trees (DT), and Bayesian Models, are widely applied in real-world situations, including spam filtering, sentiment analysis, anomaly detection, and cybersecurity analysis (Sharifani and Amini, 2023). These models have revolutionized various industries by providing tools to analyze and interpret complex data, leading to significant advancements in automation and decision-making.

1.1.1 The Shift from Centralized to Distributed Learning

In recent years, the significant development of deep neural networks, with Stochastic Gradient Descent (SGD)-based update mechanisms, has laid the foundation for what is known as Deep Learning (DL). Various model architectures have been developed to fit different tasks. For instance, convolutional neural networks (CNN) are designed to assist with computer vision tasks. At the same time, Recurrent Neural Networks (RNN) and Transformer models are better suited for natural language processing tasks (Dong et al., 2021). These advancements have enabled breakthroughs in image recognition, speech processing, and autonomous systems.

The resources and services provided by tech companies such as Amazon, Meta, and Alibaba rely heavily on Cloud Computing (CC). However, centralized machine learning or deep learning in Cloud servers cannot meet the increasing complexity of tasks and neural network architectures, even with hardware acceleration (Verbraeken et al., 2020). This limitation drives the transition from CC with **centralized machine learning** to Edge Computing (EC) with **distributed machine learning**. The significant issues of CC are summarized as follows (Liu et al., 2022). First, enormous amounts of data are

distributed across various locations. Allocating this data to a central server is costly not only due to the sheer volume but also because of privacy concerns, particularly when dealing with sensitive data such as medical records or financial information. Second, central servers encounter bottlenecks as computational and storage demands surpass the capacity of centralized hardware. Additionally, there is high latency due to the long distance between the server and edge devices. Certain applications, such as Virtual Reality (VR), Autonomous Vehicles (AV), and Smart Home Automation (SHA), require low latency for timely processing. These constraints necessitate a shift towards more localized processing solutions.

1.1.2 The Role of Knowledge Reuse and Sparsification in Distributed Learning

Distributing machine learning tasks across multiple nodes, such as mobile devices and personal computers, has addressed the aforementioned challenges through collaboratively distributed learning. This approach ensures data privacy by keeping data local and performing computational tasks close to the data generation points. According to Shi et al. (2016), such methods are supported by the infrastructure of edge computing (EC), where edge devices can handle computation offloading, data caching, and query processing. Khan et al. (2019) summarized research efforts on state-of-the-art EC, highlighting aspects such as real-time application execution, big data analytics, resource management, and security issues. For instance, an effective computation offloading strategy in the mobile EC paradigm was built on a low-complexity online algorithm (Mao et al., 2016). Additionally, Luo et al. (2018) addressed task scheduling in EC by solving a latency minimization problem using dynamic programming. Furthermore, Duan et al. (2022) proposed a minimum latency scheme with migration loads to ensure quality of service (QoS) and power-aware resource management in vehicular EC systems. Therefore, efficiency, defined as minimizing system costs such as computation, communication, and resource usage, has become a significant and trending direction in research and application within the distributed computing paradigm.

The machine learning pipeline typically involves three steps: data processing, model training, and model evaluation, with each component interacting with the others to ensure efficient learning. Reducing redundancy in this pipeline is crucial for achieving efficiency in both centralized and decentralized computing environments. The concept of *reuse, introduced by Zhou (2016), involves leveraging pre-trained machine learning models or learned knowledge to efficiently adapt to new tasks without the need to build models from scratch or reprocess data extensively.* Knowledge or model *reuse* has been extensively applied in centralized tasks such as handling concept drift (Zhao et al., 2020), protecting intellectual property (Li et al., 2021c), and accelerating query inference (Hasani et al.,

2019). With the rapid increase in data and devices in EC environments, redundancy in data and models is inevitable. Limited work applies the concept of *reuse* in EC systems. A case study in EC demonstrated the computation of new models by reusing trained models to save resources (Lee et al., 2019). A service migration scheme supported by *reuse* was proposed by Nour et al. (2021). Hence, improving efficiency in distributed computing, particularly in edge computing environments, with *reuse* remains an open challenge.

Thanks to the development of end device capabilities, there is an increasing push to evolve EC into Edge Intelligence (EI), which involves supporting deep neural networks (DNNs) at the edge. EI refers to the integration of AI and ML into EC environments. One of the most well-known EI applications is Google's G-board, which trains a text prediction model collaboratively on mobile phones using the Federated Learning (FL) paradigm. Federated Learning was introduced by McMahan et al. (2017) and has emerged as a key technique in distributed computing environments. It connects *data islands* by enabling clients to train machine learning models, especially deep learning models, collaboratively. FL can be categorized into centralized FL or decentralized FL based on whether a central server coordinates the process (Beltrán et al., 2023). Regardless of the presence of a coordinator, core challenges in FL include high communication and computation costs, system heterogeneity, data or statistical heterogeneity, and privacy concerns (Li et al., 2020c). In recent years, there has been significant interest in developing efficient FL methods to address communication and computation costs. The key point is also to reduce the redundancy in the machine learning pipeline, as stated above. To reduce the redundancy in model training step, strategies to decrease the required communication rounds for convergence include innovative client selection methods (Chen et al., 2021; Yang et al., 2021; Reisizadeh et al., 2020; Wang et al., 2022b; Li et al., 2022; Lai et al., 2021; Xu et al., 2022) and improved optimizers (Wang et al., 2022a; Zhao et al., 2022; Shi et al., 2023). Communication efficiency can be achieved by reducing redundancy in model parameters, especially the size of information exchanged among clients (Konečnỳ et al., 2016; Jiang and Borcea, 2023; Jiang et al., 2022; Wu et al., 2022). Pruning, a model compression technique, enhances computation efficiency in centralized training and has shown promise in improving both communication and computation efficiency in FL. However, it has been relatively underexplored in the context of FL (Jiang et al., 2022; Li et al., 2021a; Dai et al., 2022; Bibikar et al., 2022). Challenges remain, such as the inference performance of pruned models in FL not matching that of centralized pruned models under high sparsity. Integrating pruning with other techniques to achieve both efficiency and effectiveness while addressing additional challenges remains a promising area for research. Effectiveness refers to achieving high model performance metrics (e.g. accuracy, robustness).

1.2 Thesis Statement

This thesis claims that effectiveness can be maintained and efficiency can be enhanced in collaborative distributed machine learning by leveraging knowledge reuse and sparsification techniques. Specifically:

- It demonstrates that implementing knowledge reuse and model pruning in distributed learning frameworks, such as Federated Learning, can substantially improve model inference performance.
- These improvements come without the necessity for redundant computation and excessive communication.

The findings show that reusing trained models across nodes with similar datasets significantly reduces computational redundancy and communication overhead. Additionally, model pruning effectively decreases the model size, enhancing both computational and communication efficiency. This thesis concludes that knowledge reuse and sparsification techniques are crucial for creating efficient, scalable, and robust distributed machine learning systems.

1.3 Contributions

This thesis makes significant contributions to the field of distributed and federated machine learning, particularly focusing on enhancing system efficiency and effectiveness through knowledge reuse and sparsification techniques. The key contributions are summarized as follows:

- **Advancing Distributed Learning with Knowledge Reuse:**
  - Proposed a novel framework that integrates knowledge reuse into distributed machine learning, particularly in Edge Computing (EC) environments, enabling efficient use of previously trained models.
  - Demonstrated how leveraging similarity-based model reuse can significantly reduce computational redundancy and communication overhead while preserving data privacy.
  - Validated the proposed framework through experiments, showing that knowledge reuse achieves comparable model performance to training from scratch while reducing system costs.
- **Enhancing Reusability in Distributed Learning Frameworks:**

  - Extended the foundational knowledge reuse framework by introducing advanced reusability techniques, defined as the ability of a machine learning model to be effectively adapted for diverse tasks and environments.
  - Developed methods to systematically construct reusable models, improving both communication efficiency and computational scalability in distributed systems.
  - Evaluated these techniques through rigorous experimentation, highlighting their robustness in handling data heterogeneity and their effectiveness in improving resource utilization.

- **Introducing Pruning Techniques for Federated Learning:**
  - Designed innovative pruning methods to enhance efficiency in Federated Learning (FL), focusing on reducing model size and optimizing resource usage.
  - Proposed and evaluated the FedDIP algorithm, which incorporates dynamic pruning with error feedback and incremental regularization, achieving high sparsity without compromising model accuracy.
  - Demonstrated the potential of pruning to address communication and computation bottlenecks in centralized FL, making it feasible for large-scale and resource-constrained environments.

- **Optimizing Decentralized Federated Learning:**
  - Developed the DA-DPFL algorithm, which integrates dynamic aggregation and pruning in decentralized federated learning, addressing unique challenges like data and system heterogeneity.
  - Demonstrated how dynamic aggregation and pruning can balance computational efficiency and model accuracy, enabling scalable and robust distributed learning systems.
  - Validated the proposed methods through experiments, showcasing their energy efficiency, scalability, and adaptability across various scenarios.

These contributions collectively advance the state of the art in distributed and federated machine learning, demonstrating that efficiency and effectiveness can be achieved simultaneously through innovative frameworks and techniques. Detailed technical methodologies and experimental results are presented in subsequent chapters.

1.4 Publications From This Research

The content of the chapters is derived from the following publications:

- **Chapter 3:** (Long et al., 2022) Knowledge Reuse in Edge Computing Environments. - Qianyu Long, Kostas Kolomvatsos, Christos Anagnostopoulos. Published in *Journal of Network and Computer Applications* (JNCA, IF 7.7, JCR Q1).

- **Chapter 4:** (Long et al., 2024a) Enhancing Knowledge Reusability: A Distributed Multitask Machine Learning Approach. - Qianyu Long, Kostas Kolomvatsos, Christos Anagnostopoulos. Published in *IEEE Transactions on Emerging Topics in Computing* (TETC, IF 5.5, JCR Q1).

- **Chapter 5:** (Long et al., 2023)[1] FedDIP: Federated Learning with Extreme Dynamic Pruning and Incremental Regularization. - Qianyu Long, Christos Anagnostopoulos, Shameem Puthiya Parambath, Daning Bi. Published in *International Conference on Data Mining 2023* (ICDM 2023, CORE A*).

- **Chapter 6:** (Long et al., 2024b)[2] Decentralized Personalized Federated Learning Based on a Conditional Sparse-to-Sparser Scheme. - Qianyu Long, Qiyuan Wang, Christos Anagnostopoulos, Daning Bi. Under review in *IEEE Transactions on Neural Networks and Learning Systems* (TNNLS, IF 14.3, JCR Q1).

1.5 Thesis Outline

This thesis is structured in the following manner:

- **Chapter 2:** This chapter provides the foundational background for distributed machine learning. It covers the historical context, definitions, categories, and challenges associated with distributed statistical learning and federated learning. It provides the fundamental knowledge of distributed learning and shows the importance of this thesis.

- **Chapter 3:** This chapter introduces the concept of knowledge reuse in distributed computing, particularly within Edge Computing (EC). It details the methodology for direct model reuse, including similarity score calculation and learning paradigms. Theoretical analysis and experimental results demonstrate the efficiency and effectiveness of the proposed methods.

- **Chapter 4:** Building on the previous chapter, this chapter explores techniques to enhance the reusability of models in distributed learning frameworks. It introduces advanced methodologies for model reusability, theoretical analyses, and experimental setups to validate the proposed enhancements.

[1] Code is available at:https://github.com/EricLoong/feddip
[2] Code is available at:https://github.com/EricLoong/da-dpfl

- **Chapter 5:** This chapter focuses on sparsification techniques, specifically model pruning in centralized federated learning. It presents the FedDIP algorithm and discusses its theoretical foundations. The chapter includes experimental setups and results, showing how pruning improves computational and communication efficiency.
- **Chapter 6:** Extending the discussion on federated learning, this chapter examines decentralized federated learning with dynamic pruning. It introduces the DA-DPFL algorithm and provides theoretical and experimental analysis to demonstrate its benefits in decentralized settings.
- **Chapter 7:** The final chapter summarizes the contributions of the thesis and suggests directions for future research. It recaps the major findings and discusses potential areas for further investigation based on the thesis results.

Chapter 2

Background

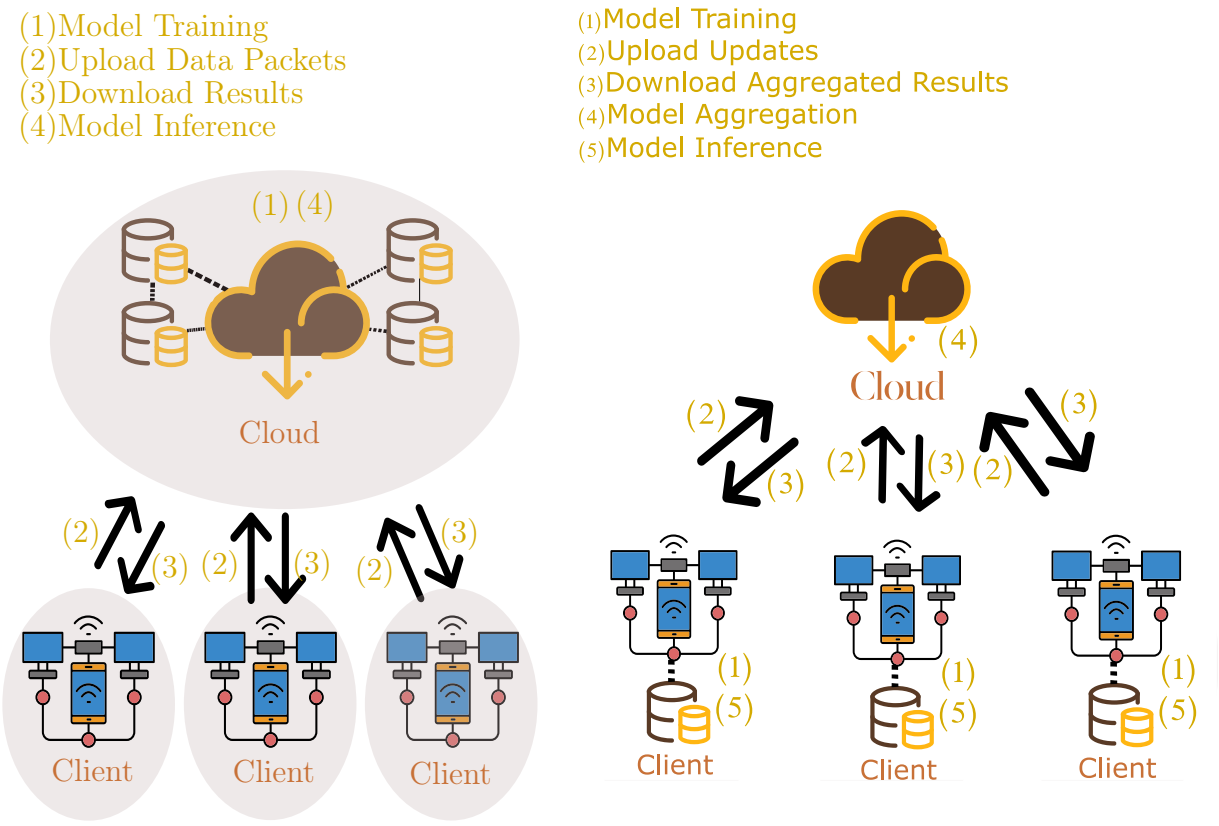
2.1 Introduction

Chapter 1 primarily discusses the motivation and challenges associated with distributed computing in the context of machine learning. To address these challenges, particularly those related to efficiency (low costs) and effectiveness (high quality of services), we have proposed leveraging the concepts of reuse with *meta*-learning and sparsification methods. This chapter overviews distributed machine learning (DML), key techniques, methods for efficiency, and current trends.

2.2 Core Concepts

2.2.1 Definitions and Scope

The section highlights the difference between cloud/centralized machine learning (CML) and distributed machine learning. Figure 2.1 explains the difference, covering both structure and function. In DML, clients generate and store their own data, reducing the need for centralized data storage and effectively addressing data distribution challenges. In contrast, CML relies on data sent from clients to a cloud server or generated by the server itself. Regarding computation, DML leverages local clients' computational power, surpassing cloud servers' limitations when handling large volumes of data. Furthermore, clients in DML can perform local inference, enabling real-time processing. Finally, while data transmission in CML is encrypted, it is more vulnerable than encrypting DML model updates. Even if encryption on updates is compromised, retrieving sensitive data from the model updates requires additional effort. Another type of distributed computing in machine learning focuses solely on accelerating training by evenly partitioning data on the server, typically referred to as *parallel computing*. Details of difference are summarized in Table 2.1. We also illustrate DML with parameter sharing in a single-step pseudo code,



(a) Cloud Machine Learning (b) Distributed Machine Learning

Figure 2.1: Comparison of Cloud and Distributed Machine Learning

as shown in Algorithm 1.

Algorithm 1 Distributed Learning with Parameter Sharing (One-Step)

Initialize model parameters ω_g
for each node i in parallel **do**
    $\omega_i \leftarrow \text{LocalTraining}(W, \text{local data})$
    Send ω_i to central server
end for
$\omega_g \leftarrow \text{Aggregate}(\omega_1, \omega_2, \ldots, \omega_i)$
for each node i in parallel **do**
    Receive updated ω_g from central server
end for

Following the classification of machine learning, Distributed Machine Learning (DML) can be broadly categorized into **Distributed Statistical Learning (DSL)** and **Distributed Deep Learning (DDL)** (Dehghani and Yazdanparast, 2023). These categories align with the underlying methods and objectives of the respective models while acknowledging their interconnected nature.

Statistical Learning (SL) focuses primarily on *interpretability*, providing clear explanations of model behavior and relationships between input and output variables. Under

Aspect	Distributed Machine Learning	Traditional Machine Learning
Data Handling	Data is distributed across multiple nodes or machines.	Data is usually centralized in a single location.
Computational Resources	Utilizes multiple computational resources to handle large-scale computations.	Relies on a single machine or a small number of machines.
Scalability	Highly scalable, can handle very large datasets and complex models.	Limited scalability, struggles with very large datasets and complex models.
Fault Tolerance	Designed to handle failures in some nodes without significant performance degradation.	Less tolerant to failures; a single point of failure can disrupt the process.
Communication Overhead	Requires significant communication between nodes for synchronization and parameter sharing.	No communication overhead since computations are localized.

Table 2.1: Differences between Distributed Machine Learning and Traditional Machine Learning

certain assumptions, SL methods can yield analytical solutions, well-constructed confidence intervals, and explicitly interpretable parameters. For example, decision boundaries in classification tasks can often be directly derived from the model. According to Hastie et al. (2009), a foundational reference for SL, statistical learning models include techniques such as *Regression, Kernel Methods, Support Vector Machines (SVM), K-Nearest Neighbors, Decision Trees, Random Forests, Ensemble Learning, and Undirected Graphical Models.* These models are typically simpler than deep learning architectures, allowing for greater efficiency in both computation and communication, particularly in distributed settings. In the context of DSL, these statistical models are adapted to operate on data distributed across multiple locations or devices.

Deep Learning (DL), on the other hand, leverages multi-layered neural network architectures to model complex, high-dimensional, and unstructured data. DL models, often considered "black-box" methods, excel at tasks like image and speech recognition, natural language processing, and other domains requiring hierarchical feature learning. Unlike SL, which emphasizes interpretability, DL focuses on representation learning and scalability. Despite this distinction, DL is inherently a subset of statistical learning, as it relies on statistical principles for optimization and prediction, a relationship explored in depth by Bartlett et al. (2021). Examples of deep learning models include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Transformer models, and Generative Adversarial Networks (GANs).

In Distributed Deep Learning (DDL), these deep learning models are trained across multiple nodes or clients to handle large datasets and complex tasks efficiently. This is achieved through distributed architectures that partition both data and model computation, enabling parallelism. Two key forms of parallelism in DDL are:

- **Data Parallelism:** The same model is trained on different subsets of data across multiple nodes, with periodic synchronization to update global parameters (e.g., federated learning).
- **Model Parallelism:** Different parts of the model are trained simultaneously on different nodes, which is beneficial for large models that cannot fit into a single device's memory (e.g., split learning).

This thesis focuses on **data parallelism** within DDL, leveraging techniques like federated learning to address challenges such as communication overhead, data heterogeneity, and privacy preservation in distributed systems.

While SL and DL are often discussed separately, they are closely related and complementary. For instance, neural networks, often associated with DL, were historically described as statistical models (e.g., perceptrons and SVMs) before evolving into deep architectures through advances in optimization and computational resources. This relationship underscores the importance of acknowledging the theoretical connections between the two paradigms while leveraging their unique strengths in distributed learning.

2.2.2 General Formulation for DML

As claimed in Chapter 1, DML is developed to support the computation across distributed multiple computational nodes or clients. Considering M nodes in a data-distributed system, each node possesses a local dataset $\mathcal{D}_i$ with $i \in \{1, 2, \ldots, M\}$. The target is to optimize an optimization function, traditionally, which is a global loss function $F(\boldsymbol{\omega}_g)$ and defined as

$$F(\boldsymbol{\omega}_g) = \frac{1}{M} \sum_{i=1}^{M} F_i(\boldsymbol{\omega}_i) \tag{2.1}$$

, where $\boldsymbol{\omega}_g$ is the global model and $\boldsymbol{\omega}_i$ is the local model for client i.

Each local loss function $F_i(\boldsymbol{\omega}_i)$ is typically of the form:

$$F_i(\boldsymbol{\omega}_i) = \frac{1}{|D_i|} \sum_{j \in \mathcal{D}_i} \ell(\boldsymbol{\omega}_i; \mathbf{x}_j, \mathbf{y}_j) \tag{2.2}$$

where $\ell(\boldsymbol{\omega}_i; \mathbf{x}_j, \mathbf{y}_j)$ is the loss incurred by the model $\boldsymbol{\omega}_i$ on the data point $(\mathbf{x}_j, \mathbf{y}_j)$.

The efficiency of a distributed learning system is influenced by the connection topology of the nodes or clients, particularly in terms of how model updates are propagated. As

noted by Verbraeken et al. (2020), different topologies exhibit varying degrees of distribution. Detailed discussions on these topologies are provided in Chapter 6. In Chapters 3 and 4, we focus on a fully connected, peer-to-peer topology to select target nodes for model reuse. Chapter 5 adopts the FedAvg topology, which is a star topology with a central parameter server. In Chapter 6, we extend the static decentralized peer-to-peer topology by allowing connections to change randomly at different rounds, enhancing the dynamic nature of the network.

2.3 Key Techniques

2.3.1 Distributed Statistical Learning

In addition to the advantage stated in Section 2.2, i.e., 'interpretability,' statistical learning also possesses another benefit, which is efficiency due to the simplicity of model structures compared with deep neural networks. However, the simple models used in statistical learning often struggle with capturing complex patterns and interactions within the data that deep neural networks excel at modeling. This limitation can result in lower predictive performance on tasks that require the learning of intricate relationships, such as image and speech recognition, natural language processing, and other domains where data complexity and volume are high. A wider understanding of distributed SL can not only be explained as in Section 2.2 but also be that leverages statistical knowledge in learning models, which are widely used in finance (e.g., predicting stock prices), healthcare (e.g., estimating disease progression), marketing (e.g., sales forecasting) Hastie et al. (2009).

Statistical knowledge informs representation learning, particularly the meta-features and meta-models learned in *meta-learning*, which is a broader concept than transfer learning (Vanschoren, 2018). Meta-features include the number of samples, features, and classes; skewness, correlation, and eigenvectors from variance decomposition; and advanced metrics like Hessian information, Fisher Information, Kullback-Leibler (KL) Divergence, and Maximum Mean Discrepancy (MMD). One approach in meta-model learning is ranking the top-K meta-models and selecting the best or multiple models fitting the task data distribution, akin to ensemble learning, to enhance overall performance. Additionally, to explain the *interpretability* of statistic models, we take *regression* as an example. The regression models provide apparent insights into the relationship between dependent ($\mathbf{Y}$) and independent variables ($\mathbf{X}$) under certain assumptions, where $(\mathbf{X}, \mathbf{Y})$ is defined in the following equation. For example, Generalized Linear Models (GLM) are interpretable because they extend linear models to accommodate non-linear relationships via the link function while retaining the linearity in the parameters $\boldsymbol{\beta}$. The general form of a GLM can be expressed as follows. The expected value of the response variable $\mathbf{Y}$ given the

explanatory variables $\mathbf{X}$ is:

$$\mathbb{E}(Y \mid \mathbf{X}) = \mu = g^{-1}(\boldsymbol{\gamma}) \tag{2.3}$$

where the linear predictor $\boldsymbol{\gamma}$ is given by:

$$\boldsymbol{\gamma} = \mathbf{X}\boldsymbol{\beta} \tag{2.4}$$

The variance of Y is a function of its mean:

$$\mathrm{Var}(Y \mid \mathbf{X}) = V(\mu) \tag{2.5}$$

According to the concrete explanation of statistical learning, we list the research of DSL, which offers *interpretability* and *efficiency* (computation, communication, and inference) in distributed systems. Navia-Vazquez et al. (2006) introduce a distributed SVM aimed at reducing information exchange between nodes while providing privacy-preserving mechanisms. Dobriban and Sheng (2021) propose an algorithm for data parallelism with distributed linear regression averaging, using one-step and iterative weighted parameter averaging. Han and Liu (2016) achieve efficient distributed statistical estimation with bootstrapping, reducing bootstrap noise variance with a weighted M-estimator method. To lower computation costs, Yao et al. (2018) develop a two-stream model under MMD constraints to achieve convergence with fewer communication rounds.

2.3.2 Distributed Deep Learning

Distributed deep learning refers to conducting learning across clients with deep neural networks to handle complex tasks (Tang et al., 2020; Verbraeken et al., 2020), particularly in tasks involving high-dimensional and unstructured data. Depending on the property of parallelism, DDL can be classified as split learning (Vepakomma et al., 2018) and federated learning (McMahan et al., 2017), which targets model parallelism and data parallelism, respectively. The pseudo-code of FedAvg (McMahan et al., 2017) is provided in Algorithm 2, which is aligning with the framework depicted in Figure 2.1b. Notably, in federated learning (FL), data is distributed across different nodes, allowing each node to train the same model architecture on different subsets of data. In the context of distributed training or parallel computing, such data partition is usually even. In contrast, FL originally aimed at training on heterogeneous data distribution. Precisely, FL can be further classified into vertical FL, horizontal FL, and federated transfer learning, depending on the heterogeneity of feature space and sample IDs. The objective of split learning is model parallelism, which splits the model across different nodes, each responsible for a part of the model. This benefits from the orchestration of the structure of neural networks and the SGD-based update mechanism.

Algorithm 2 Federated Averaging (FedAvg)

Input: M, Number of clients; T, Number of communication rounds; η, Learning rate; C: Fraction of clients to be selected each round
Output: Global model parameters
Initialization: Initialize global model parameters $\omega 0$
for each communication round $t = 1, 2, ..., T$ **do**
  Select a random fraction C of the M clients (denoted as S_t)
  Distribute the global model parameters ω^t to the selected clients in S_t
  for each client i in S_t **do**
    Client i receives the global model parameters ωt
    Client i performs local training:
    for each local epoch $e = 1, 2, ..., E_l$ **do**
      Update local model parameters ω_i^t using local data D_i and learning rate η
    end for
    Client i sends the updated local model parameters ω_i^t back to the server
  end for
  Server aggregates the received local model parameters:
  $\omega^{t+1} \leftarrow \frac{1}{|S_t|} \sum_{i \in S_t} \omega_i^t$
end for
Return: Global model parameters ω^T

Both FL and split learning allow collaborative model training without sharing raw data, addressing privacy concerns, and introducing challenges related to computation, communication efficiency, and system and data heterogeneity. Data heterogeneity refers to the variations in data distribution across different nodes, which can impact model performance and generalization. Communication overhead is a significant challenge in distributed systems, as frequent synchronization between nodes can lead to latency and increased resource usage. Computation efficiency is particularly vital in resource-constrained environments. The system heterogeneity results in difficulty coordinating computation; for example, the *stragglers* have a huge negative impact on the training efficiency, not only on the sequential training in split learning or the parallel training in FL. Furthermore, there is also a need for advanced privacy techniques against adversarial attacks.

Split Learning has emerged as a promising branch of Distributed Machine Learning (DML). Thapa et al. (2022) analyzed the relationship between FL and split learning, claiming that split learning provides a higher degree of privacy and feasibility of deployment on resource-constrained devices. Wu et al. (2023) propose a Cluster-based Parallel Split Learning (CPSL) method, which reduces training latency by partitioning devices into clusters for parallel training and then sequentially aggregating the models, coupled with a resource management algorithm that optimizes cut layer selection, device clustering, and radio spectrum allocation to adapt to wireless networks. Samikwa et al. (2022) present Adaptive REsource-aware Split learning (ARES) for efficient model training in the Internet of Things (IoT) systems that specifically deal with various operational conditions

and resource heterogeneity.

One of the particular interests of this thesis is FL, and hence we expand the discussion in Chapter 5 and 6.

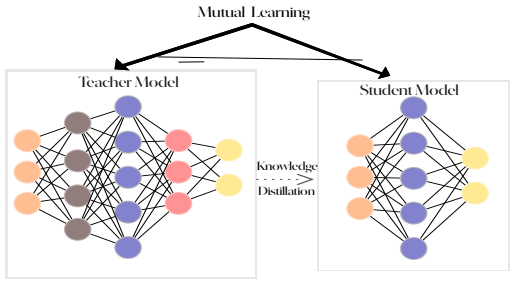
2.4 Methods For Efficiency

Achieving efficiency in machine learning systems, both distributed and centralized, involves tackling challenges such as resource utilization, latency, and scalability. A variety of state-of-the-art (SOTA) methods have been explored to address these challenges. Caching and prefetching techniques are widely used to reduce the bottleneck associated with repeatedly accessing large datasets by storing most frequently used data closer to the nodes. Task offloading and delegation methods dynamically transfer computational workloads from resource-constrained devices to more capable nodes or cloud resources to balance system efficiency. Learning rate optimization strategies, such as those used in momentum-based optimizers, improve convergence speed by adjusting the learning process. Fault tolerance mechanisms ensure efficient recovery and continued operations in distributed systems, minimizing resource waste due to interruptions or failures.

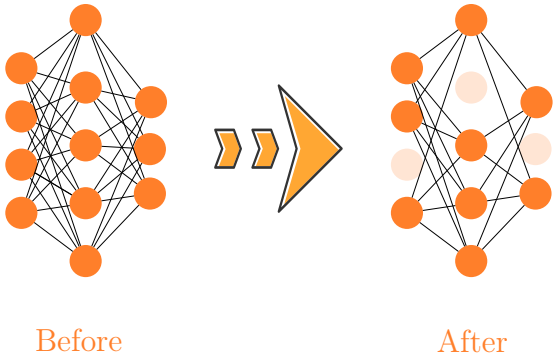
Among the various methods for improving efficiency, knowledge reuse and model compression techniques are particularly noteworthy due to their ability to address the distinct challenges inherent to distributed learning systems. Distributed learning environments frequently encounter high communication costs, primarily resulting from the repeated synchronization of large model updates across participating nodes. Additionally, computational redundancy arises when similar datasets are independently processed and trained on by multiple nodes. These issues are especially pronounced in distributed settings, where clients often operate under constraints such as limited bandwidth and restricted computational resources. By directly targeting these bottlenecks, knowledge reuse and model compression emerge as promising solutions to enhance both the scalability and practicality of distributed machine learning frameworks.

2.4.1 Compression Techniques

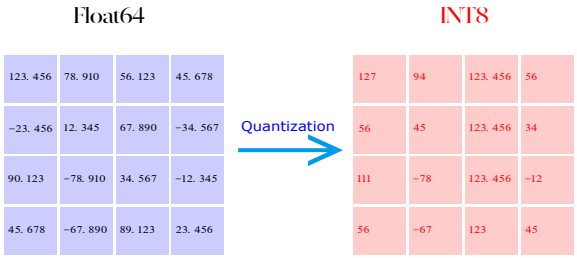
Model compression techniques such as weight pruning, quantization, low-rank factorization, and knowledge distillation, shown in Figure 2.2, have primarily been applied in centralized learning (Cheng et al., 2017). In this section, we discuss the definitions and current trends of these techniques individually.



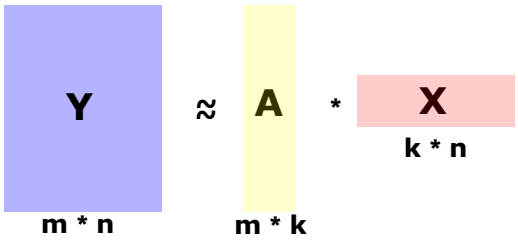
(a) Knowledge Distillation



(b) Pruning



(c) Quantization



(d) Matrix Factorization

Figure 2.2: Compression Techniques in Machine Learning

Pruning

Pruning reduces the size and complexity of models by removing redundant structures based on the importance of model components, such as neurons, connections, and layers. Figure 5.2 illustrates the differences between a neural network before and after pruning, focusing on neuron pruning. It is evident that the number of connections between neurons decreases, which reduces computational demands during training and inference, thereby enhancing efficiency.

Various methods have been proposed to optimize pruning strategies further. SNIP (Lee et al., 2018) introduced a method for pruning DNN models based on identifying important connections before training. He et al. (2017) proposed a two-step pruning method for DNN layers using regression-based channel selection and least squares reconstruction. employed the Alternating Direction Method of Multipliers (ADMM) for centralized pruning of CNNs. Following this, Lym et al. (2019) proposed PruneTrain, which utilizes structured group-LASSO regularization to accelerate CNN training in a centralized setting. The method called dynamic pruning with error feedback (DPF) was innovated by Lin et al. (2020),

which dynamically managed model sparsity through a feedback mechanism that reactivates pruned weights.

Quantization

Quantization, consisting of both post-training and training approaches, reduces storage and computational requirements by representing model weights in lower precision (measured in bits). Specifically, achieving b-bit precision means that only 2^b quantization levels are available for the model. Typically, quantization reduces the original model from 64-bit floating-point precision to 8-bit precision, as exemplified in Figure 2.2c.

This significant reduction in bit precision not only decreases the model size but also enhances computational efficiency, making it particularly advantageous for deployment in resource-constrained environments. Wu et al. (2016) introduced a framework for quantizing convolutional and fully-connected layers of Convolutional Neural Networks, achieving up to 20 times compression rate with minimal accuracy loss. This approach enables efficient inference on mobile devices by reducing memory and storage requirements. Building on these principles, Wang et al. (2018a) proposed a Two-Step Quantization (TSQ) method, optimizing both weight and activation quantization to maintain network performance while further decreasing the bit-width. Similarly, Jacob et al. (2018) developed a quantization scheme tailored for integer-arithmetic-only inference, which enhances the latency-accuracy tradeoff on mobile hardware, particularly for models like MobileNets. Lastly, Dong et al. (2019) presented HAWQ, utilizing Hessian-aware quantization to assign different bit-widths across a neural network, balancing precision and computational efficiency. These advancements underscore the critical role of quantization in enhancing the feasibility of deploying DNNs in practical, low-resource environments.

Matrix Factorization

Matrix factorization, or low-rank factorization techniques, are pivotal in compressing neural networks, significantly improving efficiency and reducing computational demands. As presented in Figure 2.2d, the original matrix $\mathbf{Y}$ is decomposed into two smaller matrices, $\mathbf{A}$ and $\mathbf{X}$, with an error matrix often omitted in the plot. This decomposition reduces dimensionality, thus achieving greater efficiency. In some cases, the rank k can be as small as 1. The compression rate for such a decomposition is given by $\frac{m \times n}{k \times (m+n)}$, as demonstrated in Figure 2.2d.

Building on the foundational benefits of matrix factorization, several researchers have proposed advanced methods to enhance compression and performance. Liebenwein et al. (2021) investigated optimal layer-wise decomposition strategies, identifying the best factorization methods to maximize both compression and performance. Yang et al. (2020) introduced a method for learning low-rank deep neural networks by enforcing singular vec-

tor orthogonality and sparsifying singular values, resulting in more compact and efficient models. Swaminathan et al. (2020) proposed sparse low-rank factorization, effectively reducing parameters while maintaining model accuracy. Additionally, Ruan et al. (2020) developed an efficient decomposition and pruning scheme (EDP) that integrates factorization with pruning, achieving substantial compression in convolutional neural networks. Idelbayev and Carreira-Perpiñán (2021) explored optimal matrix shapes and decomposition schemes for neural network compression, providing insights into the most effective strategies for various architectures. Furthermore, Idelbayev and Carreira-Perpinán (2020) focused on learning the rank of each layer in neural networks, enabling adaptive and optimal low-rank compression tailored to the specific needs of each layer.

Knowledge Distillation

Knowledge distillation is a powerful technique for shrinking neural networks by transferring knowledge from *big*-size teacher model to *small*-size student model through mutual learning, as depicted in Figure 2.2a.

This technique has been extensively studied and refined by various researchers. Hinton et al. (2015) introduced the concept of distilling knowledge from a large, redundant model into a smaller, more efficient one by using the soft targets generated by the larger model to train the smaller one. This approach allows the smaller model to retain much of the performance of the larger one while being significantly less resource-intensive. Building on this, Li et al. (2020a) proposed a block-wise supervised neural architecture search with knowledge distillation, optimizing network architectures with a teacher model's guidance, improving efficiency and model performance. Similarly, Li et al. (2020b) developed a few-sample knowledge distillation approach that achieves efficient network compression using only a limited number of training samples, addressing the challenge of data scarcity. Further advancements include Zhu et al. (2018), who presented an on-the-fly native ensemble (ONE) method for online knowledge distillation, constructing a strong teacher model during training, enhancing training efficiency and model generalization. Finally, Zhang et al. (2021a) explored self-distillation, where a model learns from its own predictions to improve efficiency and compactness, avoiding the need for a separate teacher model and thereby reducing the training overhead. These methods highlight the versatility and effectiveness of knowledge distillation in creating compact, efficient neural networks without sacrificing performance.

2.4.2 Knowledge Reuse in Machine Learning

The *learnware* concept, introduced by Zhou (2016), aims to reuse models stored in a pool of well-established models to adapt to incoming tasks, thereby avoiding the need to

train models from scratch. We extend the concept of *reuse* defined by Zhou (2016) to provide the following definition of *knowledge reuse* in the context of machine learning. *Knowledge reuse* refers to learning how to utilize complete or intermediate results, such as models and information generated during previously executed learning processes, including meta-features defined by Vanschoren (2018), to enhance the learning process for new tasks. These enhancements can reduce the required computation or improve learning performance, i.e., efficiency and effectiveness. This concept of *reuse* can be considered a subset of meta-learning.

One prominent instance of reuse is Transfer Learning (TL), which is widely applied in various deep learning areas (Weiss et al., 2016). A key branch of TL involves fine-tuning a globally pre-trained model for source tasks to address new target personalized tasks, thus avoiding expensive data-labeling efforts (Pan and Yang, 2009). This process involves reusing the information contained in the pre-trained model (learned from the source data distribution) to enhance the learning performance of the new target model. TL becomes necessary when there is a distinct difference between the source and target data distributions. In Chapter 6, we adaptively adopt this TL idea by transferring sequential knowledge from trained to untrained clients within the same training round. Conversely, if the data distribution and feature space are assumed to be *similar* in some scenarios, one can directly reuse the trained model for other tasks. We leverage this idea with statistical learning methods to achieve efficiency in Chapter 3.

Additionally, Multitask Learning (MtL) is another example of knowledge reuse. According to Caruana (1997), MtL effectively reuses knowledge to enhance learning and performance across tasks by training models on multiple tasks concurrently and leveraging shared representations. This approach is leveraged to improve efficiency of DML in Chapter 4.

The related work on knowledge reuse in the context of distributed learning is further discussed in Chapter 3.2 and Chapter 4.2.

2.5 Notation and Definitions for Future Use

We summarized the notations from Chapter 3 to 6 in Table 2.2 &2.3 orderly. Particularly, the notations used in Chapter 3 and 4 are mainly included in Table 2.2. Table 2.3 summarizes the mainly used notations in Chapter 5 and Chapter 6. The terms "*clients*," "*nodes*," and "*devices*" are used interchangeably throughout this work since the focus is distributed learning. To avoid confusion, this work adopts the following terminology based on context:

- **Node:** Refers to general computational units within the distributed system, regardless of their specific role or physical characteristics.

- **Client:** Refers to entities contributing data or participating in the federated learning process, whether in centralized or decentralized frameworks.
- **Device:** Refers to the physical hardware, such as smartphones or IoT sensors, highlighting constraints, capabilities, or applications in edge computing scenarios.

2.6 Conclusions

This chapter has provided an overview of distributed machine learning, its core concepts, key techniques, optimization techniques for efficiency, such as pruning and reuse, and notations, which are primarily utilized in this thesis.

The subsequent chapters will delve deeper into innovative learning paradigms for efficiency under various distributed learning scenarios, highlighting their importance in advancing distributed machine learning. In addition to the structural explanation in Chapter 1, we outline the following chapter structure based on the discussed content to demonstrate the consistency of this thesis. In Chapter 3, we leverage the concept of reuse to achieve efficiency in Edge Computing (EC), avoiding redundant computation by precisely identifying the potential reusable models. Chapter 4 explores leveraging reuse through multitask learning and the meta-features learned during training, such as partial learning curves, to enhance efficiency and effectiveness in distributed learning environments. Chapter 5 adapts pruning methods from centralized machine learning to achieve highly sparse models for resource-constrained clients, benefiting fast inference and conserving computational resources during training in the context of federated learning. Chapter 6 extends pruning methods to decentralized federated learning by reusing sequential knowledge from clients with updated models within the same training round. It addresses challenges such as data heterogeneity, expensive computation, communication costs, and efficient inference while preserving privacy and being robust to single points of failure.

Table 2.2: Notations Table

Symbol	Definition
M, K	Number of nodes, number of head nodes
$\mathcal{N}_i$ or $\mathcal{G}_i$	set of node i's neighbors
n_i	Number of data points in node i's dataset
d, p	Data and PLC dimensionality
$(\mathbf{X}_i, \mathbf{Y}_i)$	Input data and label matrices at node i in $\mathbb{R}^{d\times n_i}$
$\mathcal{L}$, $\mathcal{B}$	Loaners and borrowers involved in model training and usage
$f_j(\mathbf{x}) \in \mathcal{F}_{\mathcal{A}}$	Model trained on loaner j
μ_P	Kernel mean with considering embedding $\mathbf{P}$
$\mathbf{A}$	Mapping onto randomized feature space
$k(\cdot,\cdot)$	Kernel function
$\mathbf{\Sigma}$	Covariance matrices
$\lambda_k, \boldsymbol{v}_k$	k-th largest eigenvalue and eigenvector
(α, β), (α^*, β^*)	Decision thresholds for similarity
κ	GMM performance indicator
γ	Inverse of the RBF kernel bandwidth
m_k, s_k	Real and synthetic datasets
(ν, ρ)	Percentage of SVs and novelty accuracy of OCSVM
C	Quality of analytics cut-off threshold
t, τ, τ_0	Discrete time instances
ζ_t, b_t	HW time-series smoothed values and trend
θ	Model maintenance threshold over ζ_t
$\boldsymbol{\omega}, \mathbf{W}$	Model parameter and its matrix with all nodes
$\boldsymbol{\omega}_i$, $\boldsymbol{\omega}_k^*$	Local model weight in node i, tailored model weight in cluster $\mathcal{C}_k$
$\mathcal{L}_i$	Convex loss function for node i
$\mathbf{\Omega}^{-1}$	Task relationship matrix
λ_1, λ_2	Regularization parameters
$\mathcal{S}$	Ordered index set for Partial Learning Curves
$\mathcal{C}_k$	k-th cluster with $m_k < m$ nodes, $k = 1, \ldots, K$
L	Number of bootstrapping rounds
η	Learning rate
ϵ, γ	Similarity tuning parameter, SGD error tolerance
$\hat{\mathbf{V}}_i, \hat{v}_i$	Partial Learning Curve (PLC) and its magnitude on node i
$\boldsymbol{\mu}_k, \Sigma_k$	Average PLC and covariance matrix at cluster $\mathcal{C}_k$
$\zeta_i, \zeta_{\min}$	Election Probability (EP) at node i, minimum EP
$\nu_{sim}^{(k)}$	Average tasks similarity at cluster $\mathcal{C}_k$
ξ, ξ^*	Difference in reusable model accuracy
$\mu_{in}^{(k)}, \mu_{out}^{(k)}$	Model reuse performance degradation inside/outside cluster $\mathcal{C}_k$
μ_{DC}	Sørensen-Dice coefficient

Table 2.3: Notations Table (Continued)

Symbol	**Definition**
$\mathbf{x}$	Input vector
$\|\mathbf{x}\|_p, \|\mathbf{x}\|_\infty, \|\mathbf{x}\|$	ℓ_p, ℓ_∞, ℓ_2 norm of vector $\mathbf{x}$
$\mathcal{O}(.)$	Big-O notation
$f(\mathbf{x})$	Function of vector $\mathbf{x}$
$\mathbf{v}, \nabla f(\mathbf{x})$	Gradients and Gradient of function $f(\mathbf{x})$
$\min f$	Minimum value of function f
R	Reconfiguration interval for mask updates
$\mathbb{E}[\cdot]$	Expectation
$\mathcal{F}_\mathcal{A}$	Family of parametric models in parameter space $\mathcal{A}$
$\mathbf{P}$	Probability Measurements
$\mathcal{D}_i$	Local dataset for client i
$\mathcal{C}$	Set of selected clients for training
ρ_i	Proportion of data samples of client i
$\bar{\boldsymbol{\omega}}, \tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{\omega}}_i^t, \bar{\boldsymbol{\omega}}^t$	Actual and estimated model aggregation parameters
$f_i(\boldsymbol{\omega}), f(\boldsymbol{\omega})$	Local and global loss functions
T	Number of global/training rounds
$\mathbf{e}$	Error term, the difference between the model and its sparse form.
E_l	Number of local training rounds
$\mathbf{m}, \mathbf{m}_i$	Pruning mask and mask for client i
s	Sparsity level
λ_{max}	Maximum value of regularization parameters
π_i	Random bijection mapping
$\mathcal{G}_{i(a)}, \mathcal{G}_{i(b)}$	Prior and posterior client subsets
N	Waiting threshold
$\sigma_l, \sigma_g, \sigma_p$	Bounded variance for local, global, and personalized gradients
$\mathcal{P}_{t,i}$	Selection probability of client i at time t
v_t	Voting decision for pruning at time t
δ_{pr}	Pruning threshold
δ_v	Voting threshold for pruning
b, c	Pruning interval scaling factor

Chapter 3

Efficient Distributed Learning with Direct Reuse

In the thesis statement, we claim that efficiency can be achieved by directly reusing the models at the edge. In addition to Section 2.4.2, this section reviews the concept of knowledge reuse and its importance in the distributed computing paradigm. Methods are proposed to realize efficiency by optimizing model reuse across various nodes/clients.

3.1 Introduction

The rapid growth of devices, such as sensors and computing units within the Internet of Things (IoT), has emphasized the significance of Edge Computing (EC) as an essential paradigm. For example, in smart cities, sensors on traffic signals and surveillance cameras generate vast amounts of data that require real-time processing to manage traffic and ensure public safety effectively. EC is increasingly recognized for its role in addressing the limitations of Cloud Computing by offloading tasks to EC servers (nodes) that are closer to end-user devices. Recent research has been developed to overcome the challenges of Cloud Computing by offloading the tasks to EC servers (nodes) due to the proximity of EC nodes to end-user devices. For instance, Ti and Le (2017) and Zhang et al. (2017) have dealt with managing collaborative tasks in EC networks under the perspective of energy and latency minimization.

However, EC faces significant resource management challenges in the context of Machine Learning (ML), which involves data preprocessing, model training, model adaptation, and inferential analytics. These challenges are particularly appearing during the execution of collaborative tasks such as inferential analytics and predictive modeling within tight time constraints. EC nodes often struggle to complete these tasks independently due to heavy data traffic, Quality of Service (QoS) limitations, data privacy concerns, and insufficient computational power. Moreover, the growing demand for computational tasks

often exceeds the capabilities of individual nodes, making it impractical to train models on each device independently. Given the design of EC to process tasks near end-users, it is crucial to recognize that EC nodes typically lack the computational power found in cloud servers. This limitation necessitates delegating analytics and modeling tasks to neighboring edge nodes, particularly when these nodes lack the resources to perform the required processing activities.

Guo and Hu (2018) notes that many applications frequently utilize similar input data from camera feeds and share common processing components, both within the same type of applications and across various ones. Practical applications of these modeling tasks include outlier detection, nonlinear regression, classification, etc. Specifically, identifying and detecting objects such as pedestrians, traffic lights, and barriers is crucial in applications like autonomous driving. The ongoing video data generation crucial for building object detection models often leads to potential redundancy in the collected data. For instance, onboard cameras mounted on autonomous vehicles (AVs) frequently capture similar images when these vehicles traverse identical routes under comparable conditions. Consequently, exploring the reuse of locally derived knowledge, such as trained machine learning models, is essential within the Edge Computing framework. Such reuse can significantly improve the efficiency of computational resources in developing and maintaining predictive mechanisms.

Recently, the concept of *compute and model reuse* introduced by Zhou (2016) has attracted attention since it has shown the feasibility of accelerating the computation, and model inference by reusing trained models. However, most of the research leveraged the concept of reuse, such as Li et al. (2021c); Zhao et al. (2020); Vanschoren (2018) in a centralized setup. These approaches are not directly applicable to distributed learning due to several limitations, such as the assumption of direct access to all user data in a central system, which does not hold in decentralized scenarios where data privacy and the principles of distributed and collaborative learning are prioritized. Existing studies have focused on applying this concept within distributed EC scenarios but have assumed that data is held on edge servers rather than nodes. As stated in Section 2.4.2, we define *knowledge reuse* as a broader concept encompassing *model reuse*, which specifically refers to reusing pre-trained models without the need to train them from scratch. We present the following novel contributions:

We introduce a new paradigm termed knowledge reuse for reusing entire models, which enables efficient reuse of complete computations at the network edge by edge nodes. This paradigm is organized through two primary mechanisms:

- A decision-making process utilizing pairwise statistical metrics—namely, Maximum Mean Discrepancy (MMD) and Cosine Dissimilarity—to assess the feasibility of model reuse across different nodes.

- A lightweight monitoring mechanism employs the Holt-Winters forecasting method to predict potential future violations and accordingly updates the reusable models.

3.2 Related Work

In the following, knowledge dissemination in the distributed computing paradigm is discussed, and the application of model reuse and statistical machine learning for model reuse are investigated.

3.2.1 Knowledge Dissemination in EC

Anagnostopoulos (2020) describes knowledge dissemination within Edge Computing (EC) as the process of sharing trained machine learning models, termed "knowledge diffusion." This concept, distinct from task allocation, does not necessitate every node's participation in model training to complete tasks. Such strategies utilize time-updated models and selective model engagement to support real-time predictive modeling while maintaining quality of service (QoS). In enhancing communication efficiency and QoS among collaborative nodes, Anagnostopoulos et al. (2011) and Anagnostopoulos and Kolomvatsos (2018) contributed to developing context awareness at the edge. Wang et al. (2018b) proposed a Knowledge Centric Edge Computing (KCE) framework designed to dynamically explore network structures and manage communication resources by leveraging insights derived from device-to-device (D2D) communications among mobile users. Additionally, Kolomvatsos et al. (2020) applied the optimal stopping theory to determine the optimal timing for EC nodes to exchange data synopses, optimizing local data processing and task execution efficiency.

3.2.2 Model Reuse Applications

Zhou (2016) emphasized the importance of *model reuse* by firstly introducing the concept of "learnware". Pre-trained machine learning models are designed to be reusable, evolvable, and understandable, enabling efficient adaptation to new tasks without the need for extra data or compromising data privacy.

In the centralized setting, research has explored the decision-making around knowledge reuse for specific applications, such as intellectual property protection. For instance, Li et al. (2021c) introduced ModelDiff, a test-based method for calculating deep neural network similarities using decision distance vectors to represent model behavior patterns. Similarly, Zhao et al. (2020) developed a method to adaptively adjust weights for reusing models to address concept drift based on model performance. Hasani et al. (2019) presented ApproxML, a framework that constructs approximate machine learning models for

new queries by reusing existing models, demonstrating efficient model reuse. Further, Derakhshan et al. (2022) proposed an optimization strategy to eliminate redundant data processing by materializing and reusing computational models, effectively creating a reuse algorithm that integrates various pipelines into a directed acyclic graph. Additionally, Wu et al. (2021) employed reduced kernel mean embedding (RKME), one of the statistical specifications, to evaluate the feasibility of reusing pre-trained models from the learning pool.

Model reuse shares conceptual similarities with computation offloading in edge computing, as both aim to optimize resource utilization by redistributing tasks or leveraging existing resources. However, while computation offloading primarily involves delegating tasks to more capable nodes for execution, model reuse focuses on utilizing pre-trained models to minimize the computational overhead of training from scratch. Studies have explored leveraging model reuse to boost system efficiency and adaptability across distributed computing environments. A case study in Lee et al. (2019) advocated reusing (partly or fully) the trained model among multiple users to compute a new model. Since it has the potential to significantly reduce resource utilization and accelerate task completion in edge computing systems. In order to improve the utilization of resources for newly coming tasks, Nour et al. (2021) proposed an adaptive task offloading scheme called *Whispering* for migration service with the aid of computation reuse. Moreover, Bellal et al. (2021) developed CoxNet, an efficient computation reuse architecture that improves task execution times in edge computing through innovative offloading scheduling strategies. Nour et al. (2022) applies the concept of computation reuse within a federated learning framework to address incoming tasks with minimal to no additional computation, effectively reducing redundant processing and lowering computational costs.

3.3 Methodology

As outlined in Section 3.2.2, although current studies emphasize the utility of knowledge reuse for enhancing system efficiency in edge computing environments, they are not without their limitations. Primarily, these works presuppose that data is located solely on edge servers and that end users merely submit task queries without participating in model training—an assumption that fails to consider advancements in device capabilities. Sudharsan et al. (2020); Chen and Ran (2019) claimed that modern edge devices, due to hardware improvements, are now able to train models independently, thereby preserving data privacy by processing data locally and optimizing resource utilization through task offloading to edge nodes. Furthermore, the criteria used to decide reusable models, such as the scoring system described in Bellal et al. (2021), often rely on elementary data statistics like mean values. This simplistic approach may not provide adequate meta-information

for more complex tasks.

Hence, to cope with the scenarios in which data is stored across devices, we provide one learning paradigm for fully (partially) reusing knowledge. A similarity hypothesis-driven statistical learning approach is proposed for full knowledge reuse with a lightweight model reusability monitoring mechanism for stream data. *Full knowledge reuse* is defined as utilizing an entire executed model for tasks across different nodes. In contrast, partial knowledge reuse refers to employing intermediate or immature results, such as meta-features generated during the training process before model training completion.

3.3.1 Preliminaries

The notations used throughout the chapter are shown in Table 2.2. Note that nodes can be end users or devices; hence, these terminologies are exchangeable throughout the chapter's discussion.

Consider a distributed computing environment comprising M devices or nodes, collectively denoted by the set $\mathcal{M} = \{1, 2, \ldots, i, \ldots, M\}$. These edge devices are typically connected via a network topology represented by a directed graph, $\mathcal{G}$. In this chapter, we simplify the model by assuming that the nodes involved in knowledge reuse are fully connected. This ensures connectivity when computing similarity scores, a key metric for knowledge reuse. We will further explore network topology's impact on this model in the next chapters, particularly in relation to the decentralized federated learning framework.

Each node i in this network collects data represented by $\mathbf{X}_i, \mathbf{Y}_i$, where $\mathbf{X}_i \in \mathcal{R}^{n_i \times d}$ and $\mathbf{Y}_i \in \mathcal{R}^{n_i}$. Depending on whether labels are available, the data distribution at node i could be modeled as a joint probability $\mathcal{P}_i(\mathbf{X}_i, \mathbf{Y}_i)$, a conditional probability $\mathcal{P}_i(\mathbf{X}_i \mid \mathbf{Y}_i)$, or simply $\mathcal{P}_i(\mathbf{X}_i)$ if labels are not given. This initiates the first hypothesis regarding the pairwise data similarity between the two devices:

Hypothesis 1. *Data $\mathbf{X}_i$ on device i is considered similar to $\mathbf{X}_j$ on device j if and only if $\mathcal{P}_i(\mathbf{X}_i) \simeq \mathcal{P}_j(\mathbf{X}_j)$ or $\mathcal{P}_i(\mathbf{X}_i \mid \mathbf{Y}_i) \simeq \mathcal{P}_j(\mathbf{X}_j \mid \mathbf{Y}_j)$.*

This is important since, intuitively, if two devices' data, $\mathbf{X}_i$ and $\mathbf{X}_j$, are similar, a model $f_i(.)$ trained on $\mathbf{X}_i$ is likely to perform similarly when applied to when it is applied on $\mathbf{X}_j$.

Obtaining such similarity scores requires learning from the task properties, and such scores are defined as *meta-features* in Vanschoren (2018). Several metrics can be used to measure the distance between two density distributions, such as PCA skewness (Feurer et al., 2014), Kullback-Leibler Divergence (KLD) (MacKay, 2002) and Maximum Mean Discrepancy (MMD) (Sriperumbudur et al., 2010). Our proposed methods adopt MMD and Cosine Dissimilarity (CD) and modify them to suit the privacy requirements for

distributed computing environments. The details are explained in the following chapter Section 3.3.2.

Back to the edge computing ecosystem, devices are classified into two classes, the *loaners* $\mathcal{L} \subset \mathcal{M}$ and the *borrowers* $\mathcal{B} \subset \mathcal{M}$. Loaners are responsible for locally training and building predictive models over data, and borrowers need to receive trained models from the loaners because of the inability of the training model. A node is only assigned to one role simultaneously based on its capability. Given the above, the hypothesis and corresponding problems for knowledge reuse are as follows.

Hypothesis 2 (Knowledge Reuse)**.** *Let a borrower i possess data $\mathbf{X}_i$ and a loaner j possess data $\mathbf{X}_j$. If the similarity between $\mathbf{X}_i$ and $\mathbf{X}_j$, as measured by Maximum Mean Discrepancy (MMD) or the Cosine Dissimilarity (CD) of their first principal components (1st PC), exceeds a certain threshold, the model f_j, trained on the loaner's data, can be effectively reused on the borrower's data for predictive analytics. This reuse negates the need for the borrower to train a separate model f_i while achieving comparable predictive performance.*

Problem 1 (Borrower-Loaner Matching (BLM))**.** *For borrower i and loaner j within $\mathcal{M}_i$, with datasets $\mathbf{X}_i$ and $\mathbf{X}_j$ respectively, the goal is to develop a method that is lightweight, efficient in communication, and scalable. This method should ensure, without transferring $\mathbf{X}_i$ to loaner j, that the model f_j trained locally by the loaner provides accuracy and utility comparable to any model f_i that would be trained directly on $\mathbf{X}_i$.*

The Borrower-Loaner Matching (BLM) method hypothesizes that a significant *similarity* between datasets $\mathbf{X}_i$ and $\mathbf{X}_j$ enables the reuse of the loaner's model f_j on the borrower's dataset $\mathbf{X}_i$ for targeted predictive analytics tasks. This methodology requires communication efficiency between the borrower and loaner, exchanging sufficient statistical information to determine the reusability.

Problem 2 (Model Reusability Monitoring (MRM))**.** *Given a borrower i, which has borrowed the model f_j from a loaner j in $\mathcal{M}_i$, the challenge is to establish a method to continuously assess whether the borrowed model remains effective, accurate, and useful as the data $\mathbf{X}_i$ changes over time.*

When the borrower i observes that the received model becomes ineffective due to variations in the data, a new BLM process should be initiated. This proposed approach aims to identify another new loaner whose model is suitable for reuse on the borrower's new data, thereby maintaining the efficacy and accuracy of the analytical tasks.

3.3.2 Similarity Score Calculation

This section details the meta-features as similarity scores to decide model reusability based on the data across devices, including CD and MMD.

Maximum Mean Discrepancy:

According to Tolstikhin et al. (2016), consider a probability measure P in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ and a real-valued kernel function $k(\cdot, \mathbf{x})$ defined over a dataspace $\mathcal{X}$. The measure P is embedded into $\mathcal{H}$ as the *kernel mean* μ_P, where k and P must satisfy:

$$\int_{\mathcal{X}} \sqrt{k(\mathbf{x}, \mathbf{x})\, dP(\mathbf{x})} < \infty, \quad \mu_P := \int_{\mathcal{X}} k(\cdot, \mathbf{x})\, dP(\mathbf{x}) \tag{3.1}$$

Calculating this integral directly as Equation (3.1) is impractical. In a resource-constrained environment, it is more efficient to compute μ_P using the empirical estimator:

$$\mu_P := \frac{1}{N} \sum_{\ell=1}^{n_i} k(\cdot, \mathbf{x}_\ell) \tag{3.2}$$

This approach utilizes a Monte Carlo approximation to estimate kernel values, facilitating a controlled-dimensional mapping of the feature space (Rahimi and Recht, 2007).

Given this framework, we define a distance between two datasets $\mathbf{X}_i, \mathbf{X}_j \subset \mathcal{X}$ based on their kernel means μ_i and μ_j. Using the Maximum Mean Discrepancy (MMD) metric, for two probability measures P_i and P_j embedded into $\mathcal{H}$, the MMD is given by:

$$\mathrm{MMD}(i, j) = \|\mu_i - \mu_j\|_{\mathcal{H}} \tag{3.3}$$

To maintain data privacy, the computation of kernel means is performed locally at each node. In scenarios where computational complexity is a concern, we adopt the Radial Basis Function (RBF) kernel to approximate kernel values. This method significantly reduces computational requirements, managing complexity within the dimensions D and d, where D is the dimension of the mapping feature space. According to Rahimi and Recht (2007), this approximation approach involves computational and storage complexities of $O(D+d)$.

In practice, borrower node i employs this kernel approximation to compute a statistic necessary for the Borrower-Loaner Matching (BLM) process:

$$k(\cdot, \mathbf{x}) = \phi(\mathbf{x}) \approx z(\mathbf{x}) = \mathbf{A}^\top \mathbf{x} \tag{3.4}$$

with $\mathbf{A} \in \mathbb{R}^{d \times D}$ being the mapping matrix. The borrower node i then computes its kernel mean using this approximation:

$$\mu_i := \frac{1}{N_i} \sum_{\ell=1}^{n_i} k(\cdot, \mathbf{x}_\ell) \approx \frac{1}{N_i} \sum_{\ell=1}^{n_i} \mathbf{A}^\top \mathbf{x}_\ell \tag{3.5}$$

This method ensures computational efficiency while preserving the predictive accuracy within a resource-constrained edge computing environment.

Cosine Dissimilarity on 1^{st} eigenvector

Considering the properties of eigenvectors and eigenvalues derived from data, knowledge reuse facilitates lightweight synopsis exchange between loaners and borrowers. By extracting eigenvectors and eigenvalues from a dataset $\mathbf{X}$, we focus on important features through dimensionality reduction, primarily using Principal Components Analysis (PCA). This method, by performing eigen-decomposition on the covariance matrix $\mathbf{\Sigma}$ of $\mathbf{X}$, extracts d principal components, $\{\boldsymbol{v}_k\}_{k=1}^{d}$, ordered by their corresponding eigenvalues $\{\lambda_k\}_{k=1}^{d}$ in descending order.

The largest eigenvalue and its corresponding eigenvector (PC1), $\mathbf{v}_1$, projects $\mathbf{X}$ onto the eigenspace, effectively capturing the major variation within the data. This relationship is crucial for comparing datasets $\mathbf{X}_i$ and $\mathbf{X}_j$ from borrower and loaner, respectively, by evaluating the largest eigenvalues $\lambda_{1,i}$, $\lambda_{1,j}$ and eigenvectors $\mathbf{v}_{1,i}$, $\mathbf{v}_{1,j}$. As shown in Tsai and Yang (2005), eigenvalue distributions can reveal insights about the similarities between data distributions P_i and P_j.

The Power Method (PM) is utilized to approximate the largest eigenvector by: iteratively

$$\mathbf{v}_{1,t} = \mathbf{\Sigma}\mathbf{v}_{1,t-1} = \mathbf{\Sigma}^t\mathbf{v}_0, \tag{3.6}$$

converging to the largest eigenvalue λ_1, which is approximated by:

$$\lambda_{1,t} = \frac{\mathbf{v}_{1,t}^\top \mathbf{\Sigma}\mathbf{v}_{1,t}}{\mathbf{v}_{1,t}^\top \mathbf{v}_{1,t}}, \tag{3.7}$$

The complexity of PM is $O((n+T)d^2)$, where n is the number of samples and T is the required iterative steps. This combines the cost of computing $\mathbf{\Sigma}$ and iterations. We incorporate the Fast Approximate Power Iteration Method (FAPIM) (Badeau et al., 2005) for efficiency, which reduces complexity to $O(n)$.

For borrower-loaner matching, the borrower i and loaner j compute their first principal eigenvectors $\mathbf{v}_{1,i}$ and $\mathbf{v}_{1,j}$. Then calculate the Cosine Dissimilarity (CD) to assess data similarity:

$$\mathrm{CD}(i,j) = 1 - \frac{\mathbf{v}_{1,i} \cdot \mathbf{v}_{1,j}^\top}{\|\mathbf{v}_{1,i}\|\|\mathbf{v}_{1,j}\|}. \tag{3.8}$$

3.3.3 Learning Paradigm

The chapter introduces the learning paradigms to optimize knowledge reuse within distributed computing frameworks. It focuses on *direct knowledge reuse*, assumed on the availability of pre-trained models to solve Problem 1 & 2 described in Section 2. This approach leverages existing models to build direct model deployment without the need for retraining, ideally suited for scenarios where models are maintained across certain tasks.

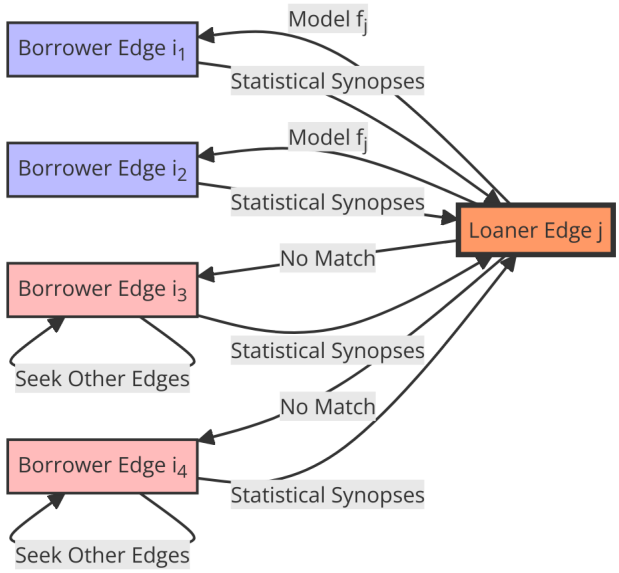


Figure 3.1: Flowchart illustrating the Borrower-Lender Matching (BLM) process between borrower edges i_1, i_2, i_3, i_4 and the central loaner edge j. This diagram depicts how statistical synopses are exchanged between the nodes to determine whether to continue or stop the BLM process based on evaluating the function $f(j)$, which assesses the reusability of shared models.

Solving BLM Problem:

Recall that in the paradigm for direct knowledge reuse, a node is assigned the roles between *loaner* and *borrower*. The decisions can be based on the connected topology, hardware capabilities, or whether the pre-trained model is stored, which is out of the scope of this chapter since we only consider a statistical learning approach for a learning paradigm. The framework for the BLM problem is illustrated in Figure 3.1, which necessitates the similarity score computation based on the statistical synopses.

In the context of calculating similarity scores between datasets from borrower and loaner nodes, as detailed in Section 3.3.2, a criterion to decide whether data is similar is essential. While the similarity score based on MMD and CD is designed for edge systems, it lacks criteria to judge the Hypothesis 1 with privacy preserved.

First, we propose to leverage the bootstrapping method to compute a test statistic, which decides whether two data are *ideally-independently-distributed* $(i.i.d)$ across two devices. The bootstrapping procedures are as follows. At loaner j, we implement a resampling strategy by randomly selecting data vectors from its dataset $\mathbf{X}_j$ with replacement to generate $K > 0$ bootstrapping datasets, denoted as $\mathbf{X}_j^{(k)}$ for $k = 1, \ldots, K$. Each dataset maintains the same size as the original dataset, with a typical setting of $K = 500$. The MMD or CD is computed for each dataset pair, leading to a sequence of bootstrap-

ping CD/MMD values. Following the method outlined in DiCiccio and Efron (1996), the threshold value α and β are calculated as

$$\alpha = \mu_{mmd} + 1.645\sigma_{mmd}, \tag{3.9}$$

and

$$\beta = \mu_{cd} + 1.645\sigma_{cd} \tag{3.10}$$

, where $\mu_{(.)}$ and $\sigma_{(.)}$ represent the mean and standard deviation of the MMD or CD values, respectively. The value 1.645 corresponds to the 95^{th} percentile of the normal distribution, thereby establishing the upper confidence boundary for a 95% confidence interval. This choice is predicated on the assumption of a one-tailed hypothesis test, thereby adopting 1.645 instead of the conventional 1.96.

Then, within the framework of Hypothesis 2, we introduce another statistical learning mechanism designed to suggest computing for the hypothesis test statistic threshold values α^* and β^* for Hypothesis 1. While sufficient data enables optimal estimation of α and β by clustering similar data distributions, practical limitations due to the scarcity of similar node-wise data and the restriction against sending real data from borrowers to loaners necessitate an alternative approach. We employ a Gaussian Mixture Model (GMM) to address these challenges to generate analogous datasets at the loaner's site. The GMM serves both as a clustering tool and a generative model that simulates the true distribution of the original data, akin to data augmentation techniques. The model's parameters, including the number of iterations and the number of Gaussian components, are optimized via cross-validation. To ease the impact on the augmented GMM-based datasets for setting threshold values, we propose model-performance-based functions that incorporate the performance of the loaner's model f_j, formulated as follows:

$$\alpha^* = e^{1+\ln(\kappa)}\mu(\text{MMD}_\text{g}) \tag{3.11}$$

and

$$\beta^* = e^{2+\ln(\kappa)}\mu(\text{CD}_\text{g}), \tag{3.12}$$

where $\kappa = \frac{Q_{GMM}}{Q_j}$ represents an elasticity parameter dependent on the relative performance of the loaner's model f_j. Here, Q_{GMM} denotes the performance of f_j tested on GMM-generated datasets, while Q_j denotes its performance on the original dataset $\mathbf{X}_j$. These equations reflect that a perfect fit of the GMM model ($\kappa = 1$) implies $\alpha^* = e \cdot \mu(\text{MMD}_g)$, signifying high confidence in reusing f_j for similar datasets. Conversely, a suboptimal GMM fit ($\kappa < 1$) necessitates stricter thresholds, scaling down α^* and β^* to reflect lower confidence in the model's reusability. This adaptive mechanism is predicated on the quality of the GMM fit, allowing for dynamic adjustment of confidence thresholds based on

empirical performance. Hence, the algorithms for solving BLM problem are included in Algorithm 3 and 4.

Algorithm 3 Knowledge Reuse Decision based on MMD

Input: M neighboring nodes ($M-1$ candidate borrowers $\mathcal{B}$ and 1 loaner); MMD similarity threshold α

Output: Set of properly identified borrowers $B_0 \subseteq \mathcal{B}$, which can receive loaner's model f_j.

```
Loaner j trains model f_j over its own data X_j.
Loaner j computes the approximation matrix A_{d×D} and kernel mean μ_j over X_j.
B_0 = ∅
for each borrower i ∈ B do
    Loaner j sends A to borrower i;
    Calculate borrower's kernel mean μ_i using (3.5);
    Borrower i sends μ_i to loaner j
    Loaner j computes MMD(i, j) using (3.3)
    if MMD(i, j) < α then
        B_0 = B_0 ∪ {i}
        Loaner j sends model f_j to borrower i for reuse.
    end if
end for
return B_0
```

Solving MRM Problem:

The proposed Model Reusability Monitoring (MRM) technique adopts the Sum of Squared Error (SSE) of the model f_j over the sliding window $\mathcal{W}_t$, providing a lightweight yet powerful metric to ensure the usability of the received model. This methodology is central to evaluating the average prediction error of f_j. As will be shown in Section 3.6, the SSE is highly correlated with Maximum Mean Discrepancy (MMD), suggesting its suitability as a substitute for MMD as the monitoring indicator in MRM.

For borrower i which utilizes the model f_j, the SSE at any time instance t is calculated as:

$$S_t = \sum_{n=1}^{n_i} (y_{n,t} - \hat{y}_{n,t})^2, \tag{3.13}$$

where $\{(\mathbf{x}_{n,t}, y_{n,t})\}_{n=1}^{n_i}$ denotes all input-output pairs in the sliding window $\mathcal{W}_t$, with predictions $\hat{y}_{n,t} = f_j(\mathbf{x}_{n,t})$.

As the sliding window updates to time $t+1$, it imports a new data pair $(\mathbf{x}_{t+1}, y_{t+1})$ while removing the oldest pair $(\mathbf{x}_1, y_1)$, leading to an updated SSE:

$$S_{t+1} = S_t - (y_{1,t} - \hat{y}_{1,t})^2 + (y_{n_i,t+1} - \hat{y}_{n_i,t+1})^2, \tag{3.14}$$

where e_1^2 and $e_{n_i}^2$ represent the squared prediction errors for the oldest and the newest

Algorithm 4 Knowledge Reuse Decision based on Eigenvector Dissimilarity

Input: M neighboring nodes (M-1 candidate borrowers $\mathcal{B}$ and 1 loaner); dissimilarity threshold β

Output: Set of properly identified borrowers $B_0 \subseteq \mathcal{B}$, which can receive loaner's model f_j.

```
Loaner j trains model f_j over its own data X_j.
Loaner j computes PC1 eigenvector v_{1,j} over X_j.
B_0 = ∅
for each borrower i ∈ B do
    Borrower i computes PC1 eigevector v_{1,i} over X_i
    Borrower i sends v_{1,i} to loaner j
    Loaner j computes dissimilarity CD(i, j) using (3.8)
    if CD(i, j) < β then
        B_0 = B_0 ∪ {m}
        Loaner j sends model f_j to borrower i for reuse.
    end if
end for
return B_0
```

pairs, respectively. The incremental update allows ΔS_{t+1} to reflect instant changes in prediction accuracy:

$$\Delta S_{t+1} = e_{n_i}^2 - e_1^2, \tag{3.15}$$

where a positive ΔS_{t+1} indicates an increase in the cumulative prediction error, thus indicating potential degradation in model performance.

MRM focuses on monitoring changes in SSE to make proactive decisions about the suitability of continuing with the borrowed model f_j. By forecasting future changes in SSE, denoted Z_t, we support the borrower with proactive decision-making regarding whether to initiate a BLM process. This is particularly crucial when the trend of Z_t at time t suggests a decline in model accuracy:

$$\tau^* = \inf\{\tau : \hat{Z}_{t+\tau} > \theta \wedge \tau_0 \leq \tau \leq h\}, \tag{3.16}$$

where τ^* identifies the earliest time the predicted error exceeds a predefined threshold θ, prompting a possible BLM initiation. We employ the Holt-Winters model for forecasting due to its efficiency in real-time scenarios, where decisions are based on the smoothed value ζ_t and trend b_t of Z_t:

$$\zeta_t = \xi_0 Z_t + (1 - \xi_0)(Z_{t-1} + b_{t-1}), \tag{3.17}$$

$$b_t = \xi_1(\zeta_t - \zeta_{t-1}) + (1 - \xi_1)b_{t-1}, \tag{3.18}$$

where the forecast at $t + h$ is $\hat{Z}_{t+h} = \zeta_t + h \cdot b_t$. This streamlined forecasting mecha-

nism ensures that decisions to initiate a BLM process are timely and based on reliable predictions.

3.4 Theoretical Analysis

The section analyzes why MMD is not suitable to be an indicator for the MRM problem and how to derive the τ^* theoretically.

3.4.1 MMD Computation with Data Stream

Evaluating the reusability of received models with respect to incoming data streams is critical. Because in Edge Computing (EC) environments, the *iid* assumption may not hold due to continuous data updates. For handling data streams, borrower nodes utilize a window-based method to maintain the most recent data for analytics. Let n_i be the sample size, we implement a sliding window mechanism as follows:

$$\mathcal{W}_{t+1} = (\mathcal{W}_t \setminus \{(\mathbf{x}_1, y_1)\}) \cup \{(\mathbf{x}_{n_i,t+1}, y_{n_i,t+1})\}. \tag{3.19}$$

This mechanism ensures only the most current data is stored in the borrower's buffer, defined at discrete time $t \in \mathbb{T} = \{0, 1, \ldots\}$, with a fixed buffer size i.

Let MMD_t be an unbiased estimator of the MMD at time t between borrower's data, $\mathbf{x}_{n,t} \in \mathcal{W}_t$, and loaner's data, $\mathbf{x}'_{n,t} \in \mathcal{W}'_t$. Then with both i number of data in the buffer, MMD_t is calculated as:

$$\begin{aligned}\mathrm{MMD}_t = \frac{1}{n_i(n_i-1)}\big(&\sum_{n=1}^{n_i}\sum_{l\neq n}^{n_i} k(\mathbf{x}_{n,t}, \mathbf{x}_{l,t}) + \sum_{n=1}^{n_i}\sum_{l\neq n}^{n_i} k(\mathbf{x}'_{n,t}, \mathbf{x}'_{l,t}) \\ - \quad &2\sum_{n=1}^{n_i}\sum_{l=1}^{n_i} k(\mathbf{x}_{n,t}, \mathbf{x}'_{l,t})\big).\end{aligned} \tag{3.20}$$

With (3.20), the incremental estimator of MMD at $t+1$:

$$\begin{aligned}&\mathrm{MMD}_{t+1} = \mathrm{MMD}_t - \frac{1}{n_i(n_i-1)} \times \\ &\big[\sum_{l>1}^{n_i} k(\mathbf{x}_{1,t}, \mathbf{x}_{l,t}) - \sum_{l>1}^{n_i} k(\mathbf{x}_{n_i,t+1}, \mathbf{x}_{l,t+1}) + \sum_{l>1}^{n_i} k(\mathbf{x}'_{1,t}, \mathbf{x}'_{l,t}) \\ &- \sum_{l>1}^{n_i} k(\mathbf{x}'_{n_i,t+1}, \mathbf{x}'_{l,t+1}) - 2\sum_{l>1}^{n_i} k(\mathbf{x}_{1,t}, \mathbf{x}'_{l,t}) + 2\sum_{l>1}^{n_i} k(\mathbf{x}_{n_i,t+1}, \mathbf{x}'_{l,t}) \\ &-2\sum_{n>1}^{n_i} k(\mathbf{x}_{n,t}, \mathbf{x}'_{1,t}) + 2\sum_{n>1}^{n_i} k(\mathbf{x}_{n,t+1}, \mathbf{x}'_{n_i,t+1})\big]\end{aligned} \tag{3.21}$$

where $\mathbf{x}_{n_i,t}$ and $\mathbf{x}'_{n_i,t}$ denotes the n_i-th data point in $\mathcal{W}_t$ and $\mathcal{W}'_t$, respectively. The incremental computation of the MMD, as shown in (3.21), can be executed with $O(n_i d)$ time complexity and $O(n_i d)$ storage requirements. However, this approach does not scale efficiently with increasing data points and dimensions for real-time applications. Additionally, this incremental calculation of MMD is impractical in scenarios where the loaner node j does not send its data to the borrower node i, thus hindering the borrower's ability to monitor the MMD evolution independently.

3.4.2 Deriviation of τ^*

To find the optimal time τ^*, we propose the following theorem.

Theorem 1. *Let $\mathcal{Z} = \{Z_1, \ldots, Z_t\}$ represent a univariate sequence of SSE changes. Given a threshold θ and the Holt-Winters coefficients (ξ_0, ξ_1), borrower i should initiate a new BLM process if and only if the expected horizon τ^*, where the sequence $Z_t(\tau)$ exceeds θ with 95% confidence, satisfies $\tau^* > \tau_0$. The horizon τ^* is determined by:*

$$\ell_t(\tau) = \tau b_t + 4.47\sqrt{\mathrm{Var}(e_t(\tau))}, \tag{3.22}$$

where $\ell_t(\tau) = \theta - \zeta_t$, and $e_t(\tau) = Z_{t+\tau} - \hat{Z}_t(\tau)$ denotes the forecasting error.

Proof. See Appendix A.1. □

3.5 Experimental Setup

We have designed three experimental scenarios to evaluate our knowledge reuse paradigm across diverse real-world datasets in Edge Computing (EC) environments. These scenarios assess the performance of our paradigm using various parametric supervised and unsupervised machine learning models. All experiments were conducted on a computer with standard computational capabilities, including an Intel Core i7 processor, 16 GB of RAM, and a GPU (NVIDIA GeForce RTX 2080).

3.5.1 Performance Metrics

We introduce *quality of analytics* metrics to evaluate the applicability of loaners' models when reused by borrower nodes. These metrics are applicable to a range of widely used parametric supervised models (such as multivariate linear and nonlinear regression) and unsupervised models (specifically for novelty detection tasks).

Supervised Learning Metrics

In the initial experimental scenarios, we focus on regression models to explore the reusability of the loaner's models. The metrics for these models include:

- Normalized Root Mean Square Error (NRMSE) for assessing prediction accuracy.
- Coefficient of Determination (R^2) for evaluating model fit.

Given a model f_j developed by a loaner on its dataset $\mathbf{X}_j$, and reused by a borrower on dataset $\mathbf{X}_i$, we define the NRMSE for the loaner's and borrower's datasets respectively as:

$$\epsilon_j = \frac{1}{\sigma_j}\sqrt{\frac{1}{n_j}\sum_{n=1}^{n_j}(y_{j,n} - \hat{y}_{j,n})^2}, \tag{3.23}$$

$$\epsilon_{ij} = \frac{1}{\sigma_i}\sqrt{\frac{1}{N_i}\sum_{n=1}^{n_i}(y_{i,n} - \hat{y}_{i,n})^2}, \tag{3.24}$$

where σ_j and σ_i are the standard deviations of the actual outputs $\mathbf{y}_j$ and $\mathbf{y}_i$, respectively.

The difference in NRMSE between the borrower's and the loaner's datasets is given by:

$$\Delta\epsilon_{ij} = |\epsilon_{ij} - \epsilon_j|, \tag{3.25}$$

which quantifies the change in prediction accuracy when the model is transferred from the loaner to the borrower.

Similarly, the coefficient of determination for both contexts is computed and the difference is:

$$\Delta R^2_{ij} = |R^2_{ij} - R^2_j|, \tag{3.26}$$

indicating the variation in model fit when applied to borrower's versus loaner's data.

Unsupervised Learning Metrics

For unsupervised learning tasks, such as novelty detection, we employ the One-Class SVM (OCSVM). This model's effectiveness in detecting outliers is quantified using the classification accuracy ratio ρ, which measures the proportion of correctly identified inliers within the boundary established by the OCSVM. The difference in classification accuracy when the model is applied to new borrower data is:

$$\Delta\rho_{ij} = |\rho_{ij} - \rho_j|, \tag{3.27}$$

where ρ_j and ρ_{ij} are the classification accuracies on the loaner's and borrower's datasets, respectively.

Note: All metrics are evaluated using a robust 10-fold cross-validation approach to ensure the reliability of our findings across diverse EC configurations. This methodology ensures that the performance differences ($\Delta\epsilon$, ΔR^2, and $\Delta\rho$) reflect genuine variations in model effectiveness due to dataset characteristics rather than random fluctuations in data sampling.

Table 3.1 provides a summary of the performance metrics utilized to assess the efficacy of the knowledge reuse paradigm per learning model and task.

Table 3.1: Performance Metrics for Knowledge Reuse

ML Model (Learning Paradigm)	**Metric**
LR (supervised; predictability, model fitting)	ΔR^2_{ij}, $\Delta\epsilon_{ij}$
SVR (supervised; predictability, model fitting)	ΔR^2_{ij}, $\Delta\epsilon_{ij}$
OCSVM (unsupervised; novelty detection)	$\Delta\rho_{ij}$

3.5.2 Scenarios Description

Experimentation Scenario I

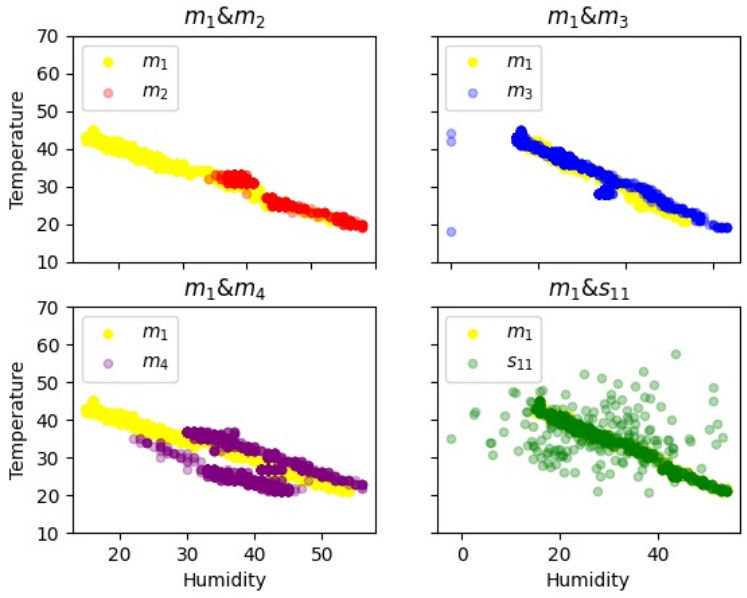


Figure 3.2: Visualization of the loaner's and borrowers' datasets of Experimentation Scenario I.

We assess our mechanisms using a real dataset[1] referenced in (Harth and Anagnostopoulos, 2018). This dataset comprises sensor measurements from four Unmanned Surface

[1]archive.ics.uci.edu/ml/datasets/GNFUV+Unmanned+Surface+Vehicles+Sensor+Data+Set+2

Vehicles (USVs) engaged in environmental monitoring on the sea surface at a naval base in Athens, Greece. Each USV, denoted as an edge node, is equipped with sensors and computational capabilities. The dataset includes four distinct sets of environmental attributes such as humidity and temperature, each corresponding to a different USV and denoted as m_1, m_2, m_3, and m_4. Figure 3.2 provides a visual representation of these datasets. Table 3.2 summarizes their characteristics.

Table 3.2: Dataset Description for Experimentation Scenario I

Detail	**Value**
Number of Real Datasets (m_k)	4
Data Dimensionality (d)	6
Total Data Points (i)	10190
Number of Synthetic Datasets (s_{kl})	32 (8 per m_k)
Machine Learning Models	LR, SVR, OCSVM
Number of Borrowers / Loaners	44 / 4 (11 borrowers per loaner)

In this scenario, we explore how our knowledge reuse mechanisms perform with datasets that exhibit inherent similarities. We generate eight synthetic datasets for each original dataset m_k, denoted as $\{s_{k1}, s_{k2}, \ldots, s_{k8}\}$. The first synthetic dataset, s_{k1}, incorporates white noise $\varepsilon \sim \mathcal{M}(0, 0.2\sigma^2_{\min})$ where $\sigma^2_{\min}$ is the minimum variance among the features. Each subsequent dataset, s_{kl} for $l = 2, \ldots, 8$, is created by blending data from m_k with a bivariate normal distribution, resulting in synthetic datasets progressively diverging from the original by replacing $(l-1) \times 12.5\%$ of m_k.

Figure 3.2 also compares the original dataset m_1 with both the borrower's original datasets m_2, m_3, m_4, and the synthetic dataset s_{11}, visually highlighting similarities and differences. Algorithms 3 and 4 are employed in this context to compute decision values (α, β) and (α^*, β^*) for identifying datasets that are identical or merely similar, respectively.

Parameter Setup: The experiments involve regression models (LR and SVR) and a novelty detection model (OCSVM). We configure the kernel parameters as follows: $\gamma_{\text{MMD}} = 0.001$ and $\gamma_{\text{SVR}} = \frac{1}{d}$, with $\nu = 0.05$ set for the OCSVM, reflecting the average outlier proportion.

This generated synthetic data tests our Hypotheses 2 under the assumption that datasets can be grouped as identical, similar, or dissimilar. For instance, dataset m_1 is presumed similar to m_3 and dissimilar to m_2 and m_4. We simulate $10,000$ synthetic datasets by resampling from $m_1, \ldots, m_4$, assigning them to nodes under varying conditions to analyze the effectiveness of our knowledge reuse algorithms. This robust simulation allows us to evaluate the practical applicability of our methodologies in real-world scenarios, supporting our theoretical claims.

Experimentation Scenario II

We extend our analysis to another real-world application by utilizing a dataset (S. et al., 2017), which comprises hourly air pollutant measurements from 12 monitoring sites in Beijing, China. The dataset features 18 attributes related to air quality. Each monitoring site's data, denoted as $\{m_1, \ldots, m_{12}\}$, is assigned to a corresponding edge node, where m_1 is designated as the loaner node and the rest as borrowers. This setup allows us to explore the feasibility of applying our knowledge reuse framework in scenarios where data similarities are implicit and not pre-defined. The loaner node builds LR, SVR, and OCSVM models, aiming for potential reuse by the borrowers based on the decision criteria derived from α^*, β^*, α, and β parameters as described in previous sections.

To adapt the dataset and the non-trivial task of evaluating similarity in a nonlinear setting, we adopt the kernel function recommendation from (Gretton et al., 2006), setting $\gamma_{\text{mmd}} = \frac{1}{\sigma_i} = 0.0014$ as justified in Section 3.3.2, where σ_i is the median value of σ. The parameters for the machine learning models remain consistent with those set in Experimentation Scenario 3.5.2.

In this experimental scenario, we lack prior knowledge about the relationships between the datasets of loaners and borrowers—whether they are similar, identical, or distinct. To facilitate a robust performance evaluation, we employ t-distributed Stochastic Neighbor Embedding (t-SNE) to project the original high-dimensional data into a two-dimensional space. This projection allows us to visually examine the datasets without distorting the inherent patterns embedded within the original data.

The performance of t-SNE is influenced by the choice of perplexity; a higher perplexity value generally improves the distinction between different datasets. To prevent overfitting and ensure a practical convergence rate, we set the perplexity to $d_P = 20$ and the learning rate to $\eta = 500$. The visualizations, as shown in Figure 3.3, enable us to discern the similarities between pairs of datasets corresponding to different edge nodes. Additionally, to illustrate the underlying correlations among the features of the datasets from various edge nodes, we present pairwise correlation matrices and associated density plots in Figure 3.4.

Table 3.3: Dataset Description for Experimentation Scenario II

Detail	**Value**
Number of Real Datasets (m_k)	12
Data Dimensionality (d)	18
Total Data Points (i)	420768
Machine Learning Models	LR, SVR, OCSVM
Number of Borrowers, Number of Loaners	11, 1 (Total $n = 12$)

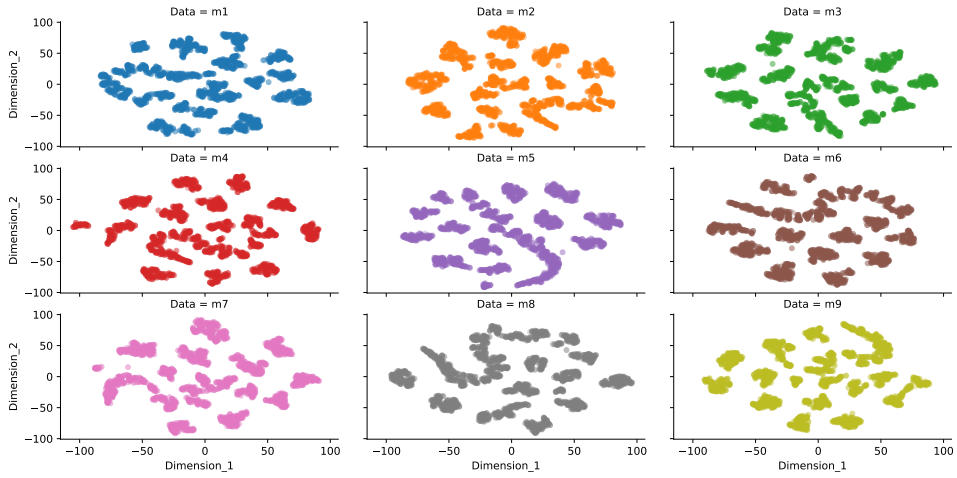


Figure 3.3: t-SNE dataset projection in Scenario II (not all datasets are visualized).

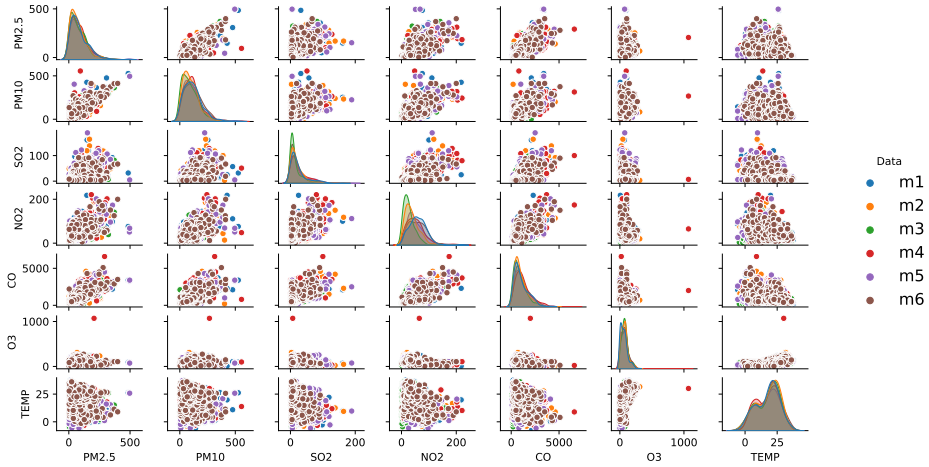


Figure 3.4: Pairwise relationships between variables in Scenario II (not all pairs of variables are visualized).

Experimentation Scenario III

To evaluate the monitoring system for the Model Reusability Mechanism (MRM) as discussed in Section 3.3.3, we analyze its performance in monitoring the Sum of Squared Errors (SSE) in relation to the Maximum Mean Discrepancy (MMD) across streaming datasets. This study utilizes a linear regression (LR) model applied to time-stamped data derived from Scenario 3.5.2.

Initially, we train the LR model using the first 200 samples from the loaner's dataset m_1, alongside the corresponding borrowers' datasets $m_{k\neq 1}$. Our experiment considers varying lengths of the sliding window $n_i \in \{100, 150, 300\}$, capturing the dynamics of incoming data and its impact on model performance. As new data entries arrive and older entries exit, we continually calculate both the instantaneous MMD (to establish a ground truth)

and the SSE for each borrower, thus verifying if the correlation between MMD and SSE accurately reflects the obsolescence of the loaner's model in the borrower's environment.

The kernel parameter $\gamma_{\text{mmd}} = 0.001$ is adopted for the MMD approximation, consistent with prior scenarios. Furthermore, to maintain robust monitoring aligned with instantaneous decision-making risks, we compute the SSE difference $\Delta S_t = e^2_{n_i,t+1} - e^2_{1,t} = e^2_{t+n_i} - e^2_t$, as defined in (3.15). We assume the model from the loaner m_1 remains constant during the ΔS computation and that both MMD and SSE are evaluated at each timestep t for comparative analysis.

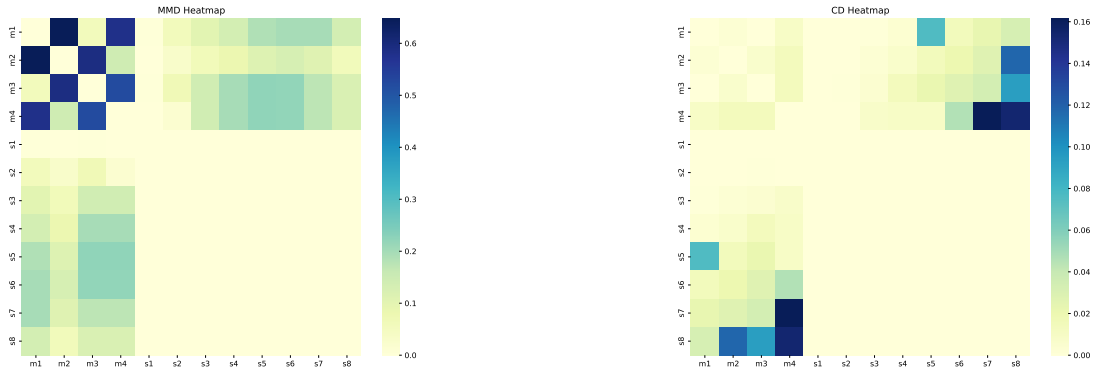
Lastly, we assess the trend effect b_t based on Z_t. Given the complexities potentially arising in solving the Holt-Winters (HW) based expected horizon τ^* as described in (3.16), we set τ_0 directly and examine whether the predicted upper bound exceeds the predefined warning level θ. This process aids in determining whether to initiate a new Borrower-Loaner Matching (BLM) process, based on the sequence $\mathcal{Z} = \{Z_1, ..., Z_T\}$, where $T = n_i = 300$.

3.6 Results and Analysis

The experimental results are analyzed with the order of scenarios. The analysis for the first two scenarios proves the usefulness of the proposed statistical learning technique for the BLM problem, and the results of the last scenario show the efficiency of the proposed monitoring mechanism for the MRM problem.

3.6.1 Analysis for Scenario I

BLM Decisions based on MMD and CD



(a) MMD Heatmap (b) CD Heatmap

Figure 3.5: Comparative heatmaps of MMD and CD metrics.

The threshold values are computed according to the proposed method in Section 3.3.3 with Equation (3.9)-(3.12), which is shown in Table 3.4 with keeping 4 decimal points.

Table 3.4: Testing Statistics for LR f_j Model

Dataset	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\alpha^*}$	$\boldsymbol{\beta^*}$
m_1	0.021443	0.000005	0.079761	0.000080
m_2	0.016278	0.000044	0.067161	0.000899
m_3	0.020115	0.000022	0.079831	0.000302
m_4	0.011548	0.001575	0.044610	0.006993

The heatmaps, shown in Figure (3.5a) and (3.5b), present a comparative assessment of $\{m_1, m_2, m_3, m_4\}$ and datasets (comprising 1 loaner and 11 borrowers as discussed in Section 3.5.2). It is important to note that CD and MMD are directionless due to our approximation to the kernel matrix $\mathbf{A}$, which depends on D and d. Notably, D equals d when considering datasets with identical features, thus ensuring $\text{MMD}(\mathbf{X}_i, \mathbf{X}_j) = \text{MMD}(\mathbf{X}_j, \mathbf{X}_i)$. This symmetry guarantees that both the loaner and borrower agree on the (dis)similarity of their datasets and the model reusability decision without further communication.

Given the MMD-/DC-dataset similarity between the loaner and borrowers, it is evident that CD increases as the amount of noisy data increases. This is because CD computes the cosine dissimilarity between PC1 for the borrower's $\mathbf{X}_i$ and the loaner's $\mathbf{X}_j$, where PC1 is a linear combination of the features that maximizes variance at the first step of the PM. With an increase in variance in a dataset $\mathbf{X}_i$, $\text{CD}(\mathbf{X}_j, \mathbf{X}_i)$ will correspondingly increase. This property of CD makes it a suitable metric for sensitively differentiating two datasets based on variance differences, thereby suggesting model reusability with high confidence, as demonstrated later.

By analyzing Figure 3.5 along with Table 3.4, we draw the following conclusions regarding the capability of the proposed BLM for reusing models across the edge network:

- According to the β estimated threshold, we conclude that only the first synthetic dataset s_{k1} is IID to its original m_k for $k \in \{1, \ldots, 4\}$, which verifies the efficiency of the statistical learning approach.

- Considering the β^* threshold, the loaner considers only s_{11} as *similar* to its dataset m_1. The datasets $\{s_{k1}, s_{k2}\}$ are *similar* to m_k for $k = \{2, 4\}$, and $\{s_{31}, s_{32}\}$ are tagged as *similar* to m_3, indicating a failure in correctly classifying the similar relationship between m_3 and m_1. However, this allows more elasticity than the condition of IID.

Given the decision thresholds α and α^*, we observe the following:

- The α threshold works, as all s_{k1} datasets are classified as *identical* to m_k as expected, while all s_{k2} are labeled as *similar* with the aid of α^*. Additionally, s_{23} is also classified as *similar* to m_2, which is acceptable.

- Utilizing α^*, the loaner can successfully detect that m_1 is *similar* to m_3, as evidenced in Figure 3.2.

Based on the above, the estimation and adoption of the decision thresholds indicate that the loaner, using the MMD and CD metrics, can successfully determine whether a candidate borrower can be provided the loaner's model for reusability based on the similarity of the corresponding datasets. This statistical-based decision-making on model reusability denotes an essential component of knowledge reuse. The next step in assessing the performance of the reused models on borrowers' data is to study the discrepancy of the quality analytics metrics introduced in Section 3.5.1.

Performance Metrics Evaluation

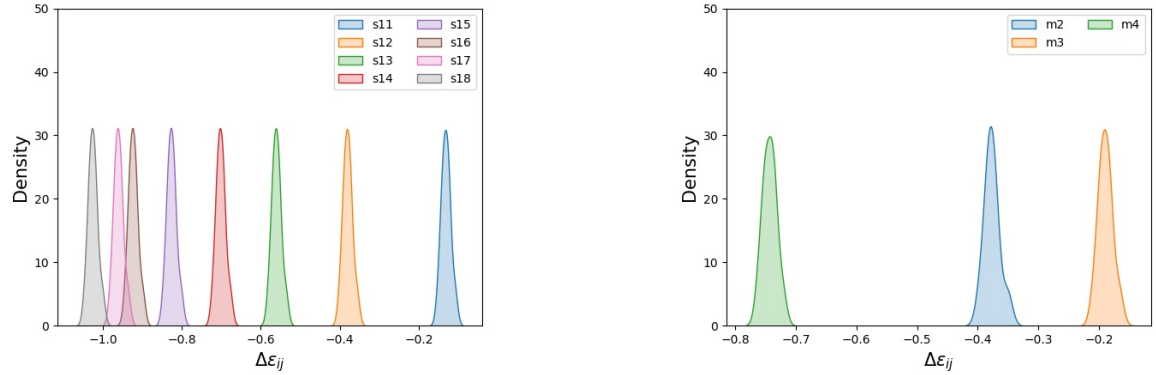
We report on the performance metrics outcomes for models built on the loaner's data when evaluated on the borrower's datasets (refer to Table 3.1). Notably, a positive expected $\mathbb{E}[\Delta\epsilon_{ij}]$ indicates that the borrower's performance exceeds that of the original loaner. Conversely, negative $\mathbb{E}[\Delta R^2_{ij}]$ and $\mathbb{E}[\Delta\rho_{ij}]$ suggest superior performance by the borrower's models.

To ensure the quality of analytics, we discourage results where $\mathbb{E}[\Delta\epsilon_{ij}] < -C$ for a specific *cut-off analytics threshold*, which varies by application and performance metric. Similarly, we do not reuse models whose $\mathbb{E}[\Delta R^2_{ij}]$ or $\mathbb{E}[\Delta\rho_{ij}]$ exceed C.

As an illustration, consider the dataset m_1 as the loaner j's dataset $\mathbf{X}_j$, while other datasets are associated with borrower nodes $\mathbf{X}_i$. To investigate discrepancies in reusing supervised regression models, we examine the model fitting difference ΔR^2_{ij} and predictability difference $\Delta\epsilon_{ij}$. Figures 3.6 and 3.7 show the distribution (density) of these metrics after reusing the loaner's LR models with successfully matched borrowers' datasets.

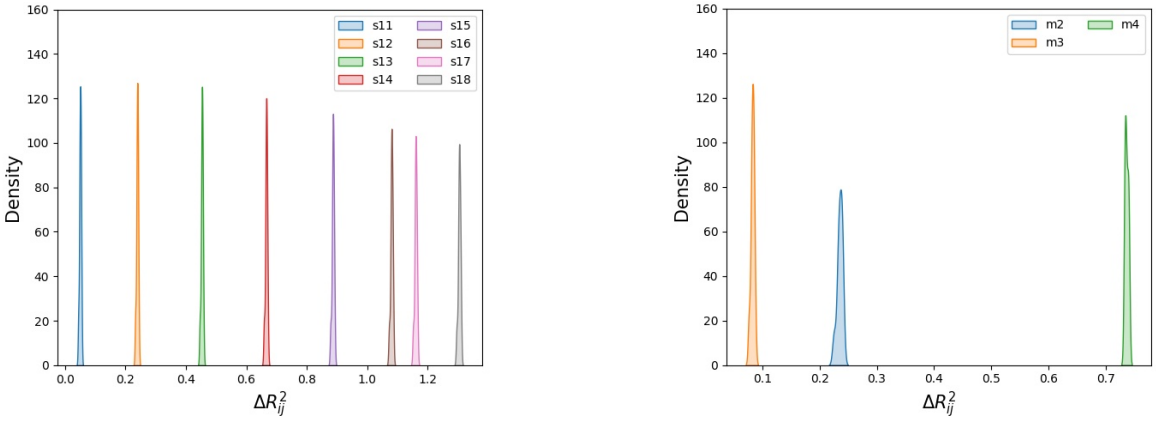
The observed patterns align with our hypothesis that reusing models over similar datasets yields satisfactory analytics quality. This confirms the applicability of the BLM process in identifying loaner-borrower pairs that can share/reuse models, achieving promising model-fitting analytics. Specifically, the distributions of s_1-s_8 show distinct separation, with performance metrics degrading as error percentages increase. Notably, m_3 demonstrates the best quality of analytics, with $\Delta R^2_{ij} \to 0$ and $\Delta\epsilon_{ij} \to 0$. We observe similar results for SVR model reusability, as evidenced in Figures 3.8 and 3.9.

For unsupervised/novelty detection models reused by matched borrowers, we report the discrepancy in novelty detection accuracy of OCSVM (refer to Table 3.1). Figure 3.10 shows that $|\Delta\rho_{ij}|$ increases with the proportion of noisy data. However, $\Delta\rho_{ij}$ for m_4 is



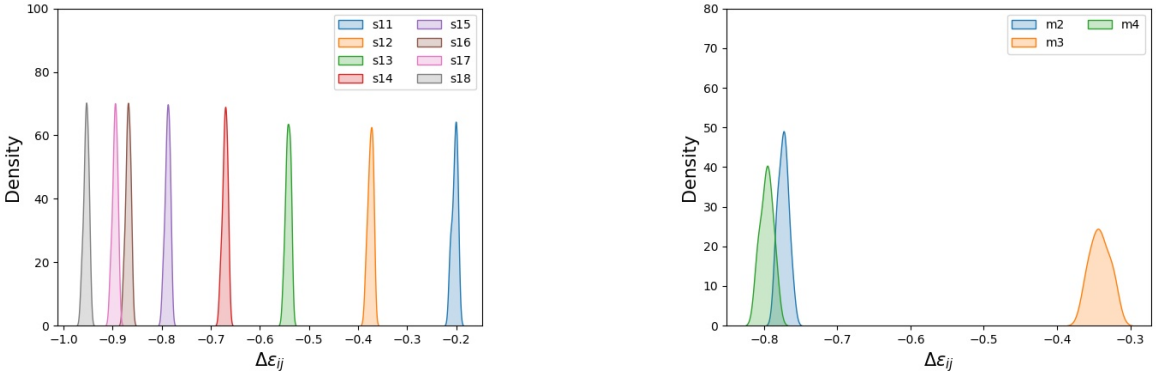
(a) (b)

Figure 3.6: Density plots of LR for $\Delta\epsilon_{ij}$ in Scenario I.



(a) (b)

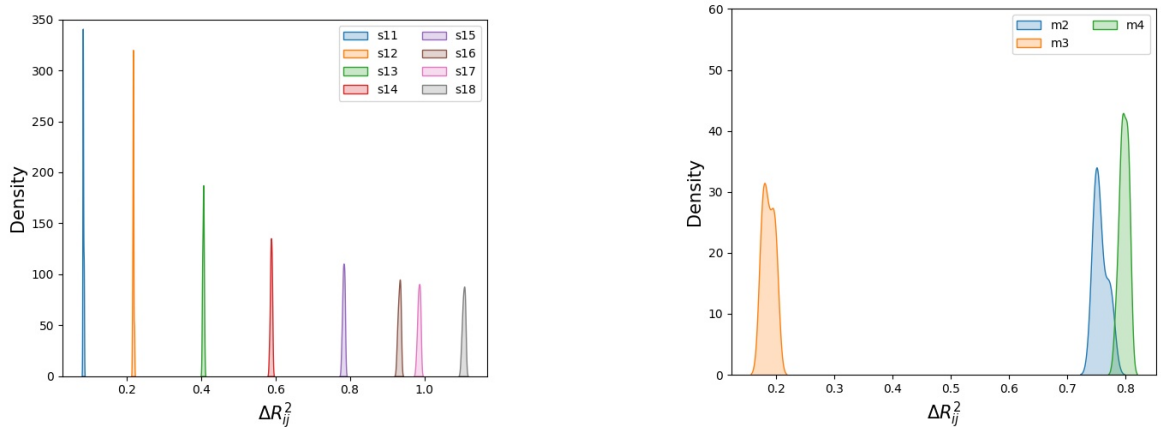
Figure 3.7: Density plots of LR for ΔR_{ij}^2 in Scenario I.



(a) (b)

Figure 3.8: Density plots of SVR for $\Delta\epsilon_{ij}$ in Scenario I.

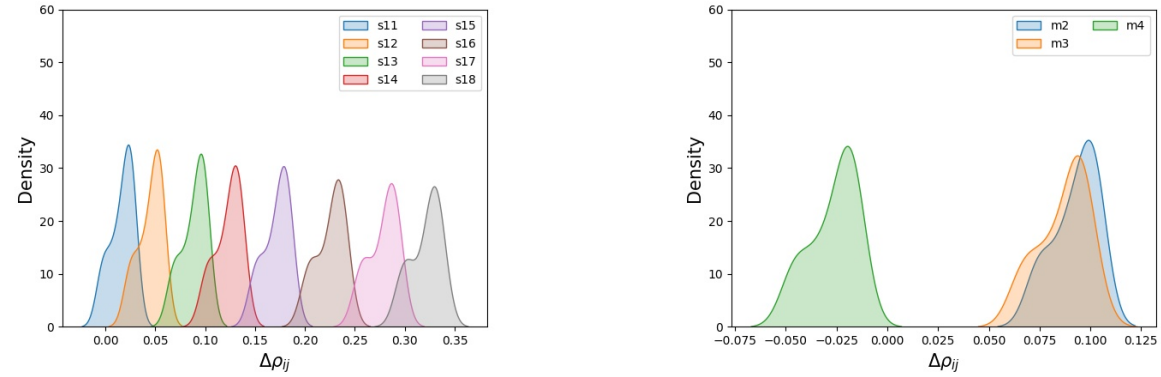
closer to 0 compared to m_3, indicating better results due to learned boundaries shown in Figure 3.11. Specifically, a $\Delta\rho_{ij} < 0$ but close to zero for m_4 suggests fewer outliers than



(a) (b)

Figure 3.9: Density plots of SVR for ΔR_{ij}^2 in Scenario I.

m_1, despite m_3 being more similar to m_1. Figure 3.11 illustrates that inlier data points (marked purple) are mostly concentrated around the red data points. Nevertheless, for similar datasets, reused OCSVM over m_4 performed better than on m_3, though $\Delta\rho_{ij} < 0.1$ for m_3.



(a) (b)

Figure 3.10: Density plots of OCSVM for $\Delta\rho_{ij}$ in Scenario I.

We evaluate the efficiency of reusability for different performance metrics given specific cut-off thresholds $C \in \{0.1, 0.2, 0.4\}$, summarized in Tables 3.5 and 3.6. The reusability efficiency is defined as the ratio of detected borrowers with similar/identical datasets under threshold values α^* or β^* to the total number of candidate borrowers suggested in set $\mathcal{B}_0$ (see Algorithms 3 and 4). NaN indicates cases where the denominator is zero.

As observed, the number of detected similar datasets remains fixed for specific α^* and β^* values. To improve efficiency, enlarging the boundaries of α^* and β^* increases the

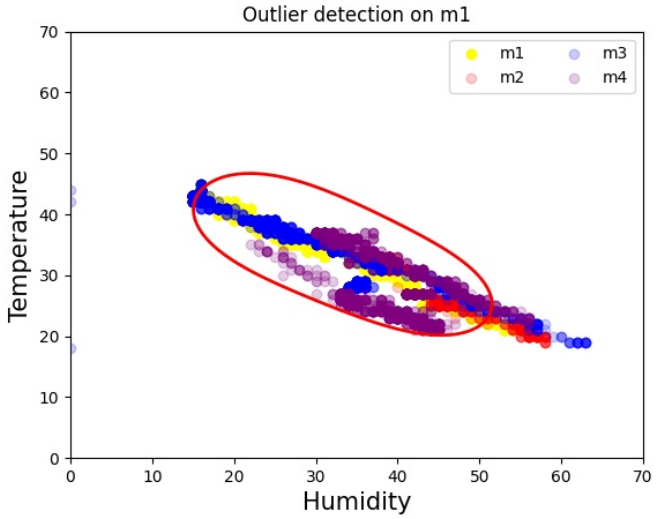


Figure 3.11: Novelty detection data-space boundary (red curve) adopting OCSVM over m_1.

likelihood of detecting more similar borrowers. False positive rates are not discussed in Scenario 1, as we focus on optimizing borrower detection satisfied with quality analytics. For instance, with $\Delta\epsilon_{ij}$(LR) and $C = 0.2$, three datasets (m_3, s_{11}, s_{12}) are considered similar, though s_{12} is a false positive as its performance metrics exceed 0.2. No false positives occur for β^*, as it only considers s_{11} similar, a white-noise version of the original source.

On average, at least 75% of borrower nodes can reuse regression models from loaners, indicating significant model reusability impact at the network edge, while up to 50% of borrower nodes can reuse unsupervised learning models.

Table 3.5: Scenario 1 (α^*): Reusability Efficiency over performance metrics vs cut-off quality threshold C.

Metrics	0.1	0.2	0.4
$\Delta\epsilon_{ij}$(LR)	NaN	100%($\frac{2}{2}$)	100%($\frac{3}{3}$)
ΔR^2_{ij}(LR)	100%($\frac{2}{2}$)	100%($\frac{2}{2}$)	75%($\frac{3}{4}$)
$\Delta\epsilon_{ij}$(SVR)	NaN	100%($\frac{1}{1}$)	100%($\frac{3}{3}$)
ΔR^2_{ij}(SVR)	100%($\frac{1}{1}$)	100%($\frac{2}{2}$)	75%($\frac{3}{4}$)
$\Delta\rho_{ij}$(OCSVM)	50%($\frac{3}{6}$)	37.5%($\frac{3}{8}$)	27.3%($\frac{3}{11}$)

Table 3.7 shows the efficiency of decision values. In this experiment, 90% edge nodes holing i.i.d data are obtained by subsampling from m_1. The α and β values successfully detect identical data distributions. With the α^* threshold, we cluster *similar* datasets more sensitively as the portion labeled as *similar* increases across 1000 edge nodes. Using α and β decision values, we achieve 90% of IID edge nodes regardless of changes in y% and z%, indicating the effectiveness of the estimated threshold values in BLM decision-making.

Table 3.6: Experiment 1 (β^*): Reusability Efficiency over performance metrics vs cut-off quality threshold C.

Metrics	0.1	0.2	0.4
$\Delta\epsilon_{ij}$(LR)	NaN	50%($\frac{1}{2}$)	33.3%($\frac{1}{3}$)
ΔR^2_{ij}(LR)	50%($\frac{1}{2}$)	50%($\frac{1}{2}$)	25%($\frac{1}{4}$)
$\Delta\epsilon_{ij}$(SVR)	NaN	100%($\frac{1}{1}$)	33.3%($\frac{1}{3}$)
ΔR^2_{ij}(SVR)	100%($\frac{1}{1}$)	50%($\frac{1}{2}$)	25%($\frac{1}{4}$)
$\Delta\rho_{ij}$(OCSVM)	16.7%($\frac{1}{6}$)	12.5%($\frac{1}{8}$)	9.1%($\frac{1}{11}$)

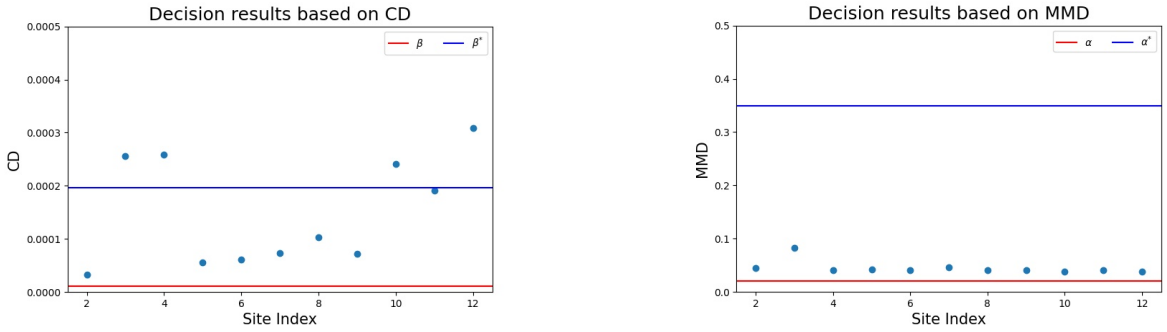
The α^* threshold is more sensitive to changes in y, which can be exploited to adjust the degree of acceptable model performance degradation. Depending on the application and desired trade-off, some dissimilar datasets may be considered *similar*. In our settings, we ensure that model performance degradation remains within $[-0.2, 0.2]$, reflecting the flexibility of the α^* threshold in balancing resource utilization efficiency and analytics quality.

Table 3.7: Percentage of similar datasets detected (x=90%)

y%	α IID (%)	α^* Similar (%)	β IID (%)	β^* Similar (%)
0	89.99	90.00	90.00	90.01
2	89.98	92.00	90.00	90.00
4	90.00	94.00	90.00	90.00
6	89.97	95.99	89.99	90.00
8	89.98	97.98	89.99	90.00

3.6.2 Analysis for Scenario II

We report on the performance of the knowledge reuse mechanism, focusing on the scenario where m_1 is the dataset of the loaner node at Site 1. Figure 3.12 illustrates the similarity measurements (CD and MMD) between the loaner's m_1 and the borrowers' datasets with decision values, which indicates that all m_k datasets ($k \in \{2, \ldots, 12\}$) are not IID to m_1.



(a) (b)

Figure 3.12: (Upper) CD and (lower) MMD decision values in Scenario II for all datasets (edge sites).

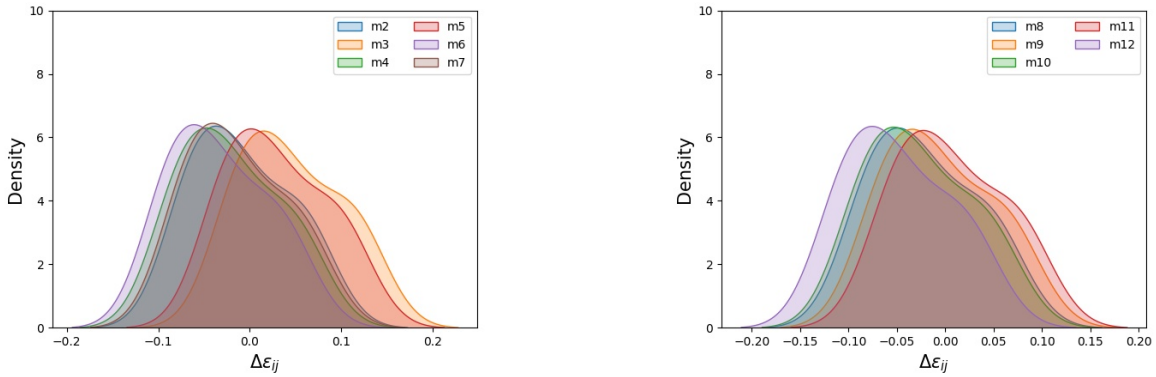
The proposed methods for determining dataset similarity do not always agree due to differing statistical synopses. Figure 3.12 shows that α^* suggests all m_k datasets are similar based on MMD, while β^* identifies $\{m_3, m_4, m_{10}, m_{12}\}$ as dissimilar. The nature of the ML models under reusability influences these decisions. For example, m_{12} has the least $\Delta\epsilon_{ij}$ in Figure 3.13, yet β^* excludes it from the candidate borrowers.

Adopting α^* may include less similar datasets like m_{12}, leading to varied reusability decisions. Figures 3.13 and 3.14 show that ϵ_{ij} for m_3 is the largest, and R^2_{ij} is the smallest, suggesting acceptable model fitting but varied predictability for LR models.

For SVR models, Figures 3.15 and 3.16 show m_6 has poor outcomes, with the smallest $\Delta\epsilon_{ij}$ and the largest ΔR^2_{ij}. Given $|\Delta R^2_{ij}| < 0.1$, all borrowers associated with $\{m_2, \ldots, m_{12}\}$ are similar to loaner's m_1, although decisions differ between LR and SVR models. Using β^* for SVR models can successfully detect dissimilar datasets such as m_{12}, ensuring higher reusability accuracy.

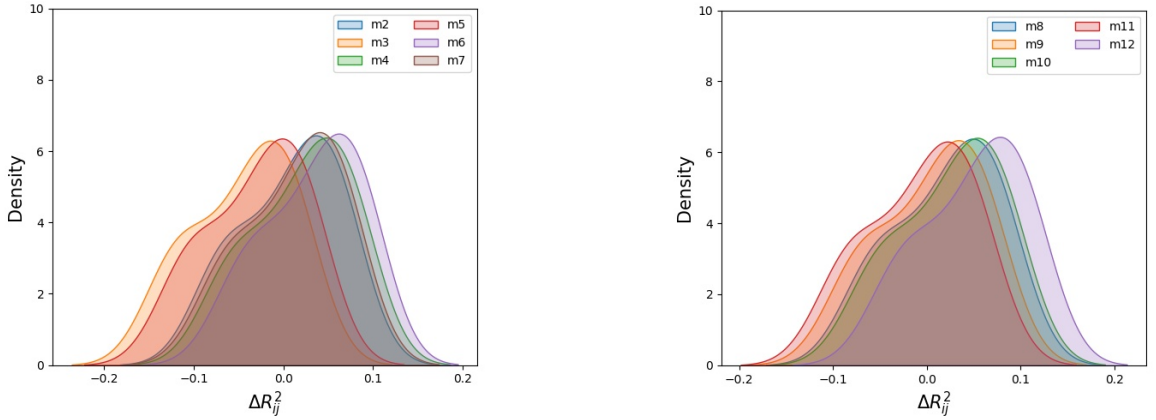
For novelty detection, Figure 3.17 shows that borrowers of m_3 and m_7 have the highest $\Delta\rho_{ij}$, correctly excluding them from the BLM process. Most $\Delta\rho_{ij}$ values for other borrowers range from 0 to 0.1, indicating effective reusability of classification models at the network edge, with 63.6% of borrower nodes reusing the loaner's model confidently.

We also examine reusability efficiency by observing performance metrics at different thresholds ($C \in \{0.05, 0.1, 0.2\}$) in Table 3.8. The number of detected similar datasets under α^* is consistently high, and we differentiate the performance of α^* and β^* based on false positive rates in Tables 3.9 and 3.10. For instance, with $C = 0.05$ for $\Delta\epsilon_{ij}(LR)$, α^* detects all similar datasets with zero false-positive rate, illustrating the robustness of the knowledge reuse paradigm.



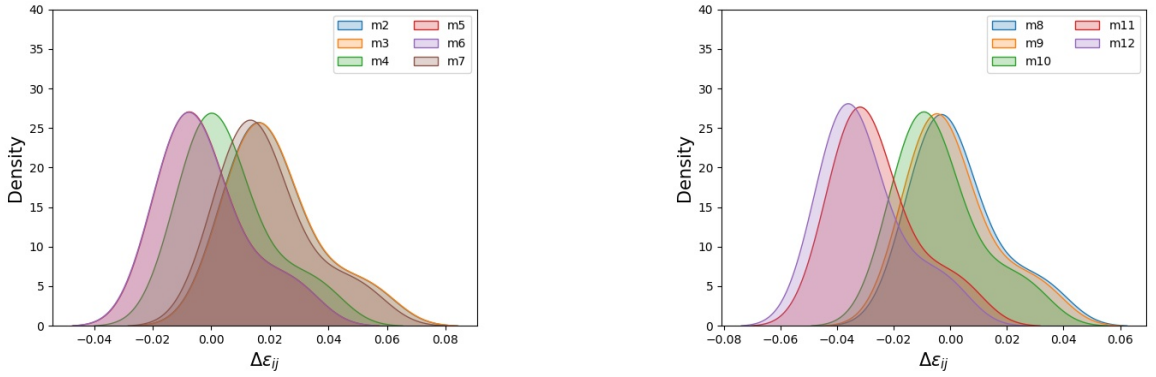
(a) (b)

Figure 3.13: Density plots of LR for $\Delta\epsilon_{ij}$ in Scenario II.



(a) (b)

Figure 3.14: Density plots of LR for ΔR_{ij}^2 in Scenario II.

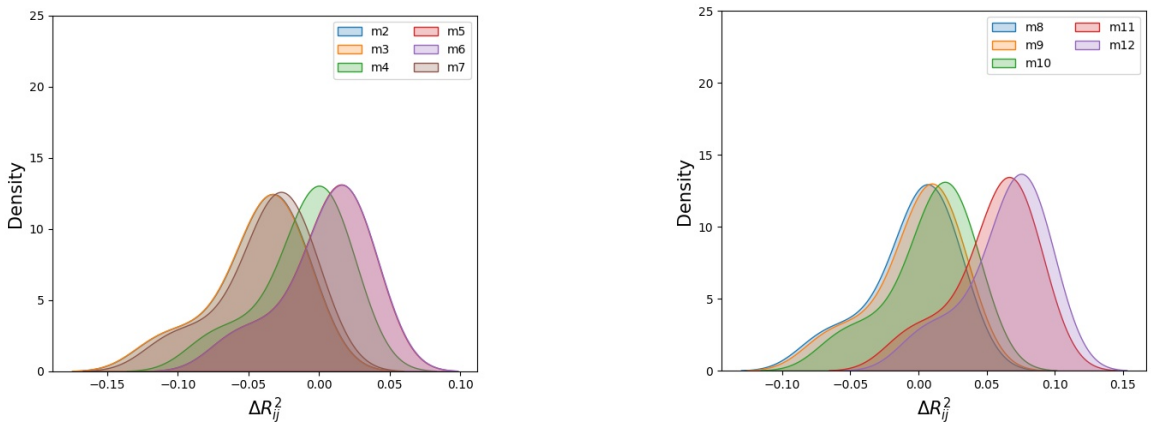


(a) (b)

Figure 3.15: Density plots of SVR for $\Delta\epsilon_{ij}$ in Scenario II.

3.6.3 Analysis for Scenario III

In this experiment, we examine the feasibility of SSE as a lightweight indicator of the Model Reusability Mechanism (MRM). We first investigate the relationship between SSE



(a) (b)

Figure 3.16: Density plots of SVR for ΔR_{ij}^2 in Scenario II.

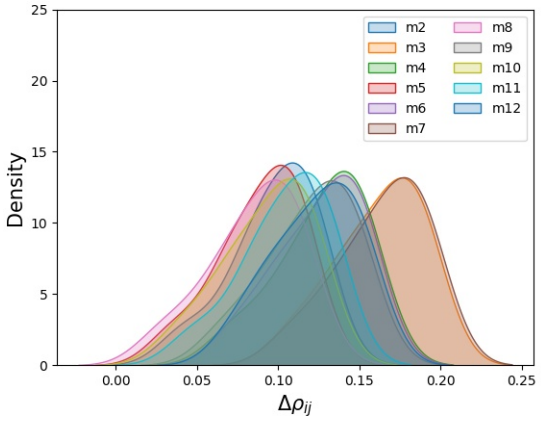


Figure 3.17: Density plots of OCSVM for $\Delta\rho_{ij}$ in Scenario II.

Table 3.8: Scenario 2 (β^*): Reusability Efficiency over performance metrics vs cut-off quality threshold C.

Metrics	0.02	0.05	0.1
$\Delta\epsilon_{ij}$(LR)	83.3%($\frac{5}{6}$)	63.6%($\frac{7}{11}$)	63.6%($\frac{7}{11}$)
ΔR_{ij}^2(LR)	85.7%($\frac{6}{7}$)	63.6%($\frac{7}{11}$)	63.6%($\frac{7}{11}$)
$\Delta\epsilon_{ij}$(SVR)	66.7%($\frac{6}{9}$)	63.6%($\frac{7}{11}$)	63.6%($\frac{7}{11}$)
ΔR_{ij}^2(SVR)	66.7%($\frac{6}{9}$)	63.6%($\frac{7}{11}$)	63.6%($\frac{7}{11}$)
$\Delta\rho_{ij}$(OCSVM)	NaN	NaN	75%($\frac{3}{4}$)

and MMD metrics. We denote the MMD between the loaner's m_1 and one of the borrowers' datasets $\{m_2, m_3, m_4\}$ as 'vs m_k'. The left results in Figure 3.18 shows that MMD is informative for guiding knowledge reuse and monitoring the quality of analytics of the reused model. Specifically, SSE is a direct performance indicator of the borrowed model, where a relatively small MMD generally corresponds to a relatively small SSE.

Table 3.9: Scenario 2 (α^*): False positive rate for different performance metrics vs cut-off quality threshold C.

Metrics	0.02	0.05	0.1
$\Delta\epsilon_{ij}$(LR)	45.5%($\frac{5}{11}$)	0.0%($\frac{0}{11}$)	0.0%($\frac{0}{11}$)
ΔR^2_{ij}(LR)	36.4%($\frac{4}{11}$)	0.0%($\frac{0}{11}$)	0.0%($\frac{0}{11}$)
$\Delta\epsilon_{ij}$(SVR)	18.2%($\frac{2}{11}$)	0.0%($\frac{0}{11}$)	0.0%($\frac{0}{11}$)
ΔR^2_{ij}(SVR)	18.2%($\frac{2}{11}$)	0.0%($\frac{0}{11}$)	0.0%($\frac{0}{11}$)
$\Delta\rho_{ij}$(OCSVM)	100%($\frac{11}{11}$)	100%($\frac{11}{11}$)	63.6%($\frac{7}{11}$)

Table 3.10: Scenario 2 (β^*): False positive rate for different performance metrics vs cut-off quality threshold C.

Metrics	0.02	0.05	0.1
$\Delta\epsilon_{ij}$(LR)	28.6%($\frac{2}{7}$)	0.0%($\frac{0}{7}$)	0.0%($\frac{0}{7}$)
ΔR^2_{ij}(LR)	14.3%($\frac{1}{7}$)	0.0%($\frac{0}{7}$)	0.0%($\frac{0}{7}$)
$\Delta\epsilon_{ij}$(SVR)	14.3%($\frac{1}{7}$)	0.0%($\frac{0}{7}$)	0.0%($\frac{0}{7}$)
ΔR^2_{ij}(SVR)	14.3%($\frac{1}{7}$)	0.0%($\frac{0}{7}$)	0.0%($\frac{0}{7}$)
$\Delta\rho_{ij}$(OCSVM)	100%($\frac{7}{7}$)	100%($\frac{7}{7}$)	57.1%($\frac{4}{7}$)

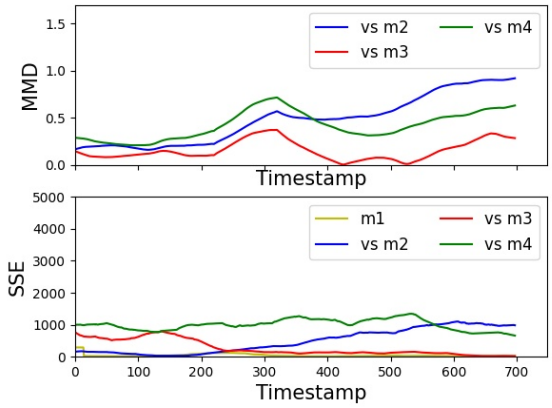
Observations indicate that the SSE of m_2 is relatively low at the beginning, with MMD at acceptable levels, suggesting that m_2 has the best model reusability among the datasets $\{m_2, m_3, m_4\}$. The SSE of m_3 is even lower than that of m_1 itself, with a relatively low MMD. While the initial MMD values of the three datasets are close, the average MMD and SSE for m_3 are the lowest over the entire time range, confirming that m_3 is the most similar to m_1.

Another pattern observed is that changes in SSE almost predict MMD changes, with increases or decreases in SSE occurring before MMD. The average values for 'vs m_2' and 'vs m_4' suggest that MMD trends coordinate with SSE trends. The time-series plot of ΔSSE in Figure 3.19 shows that m_3 closely aligns with m_1, indicating that m_3 is the best candidate for model reusability.

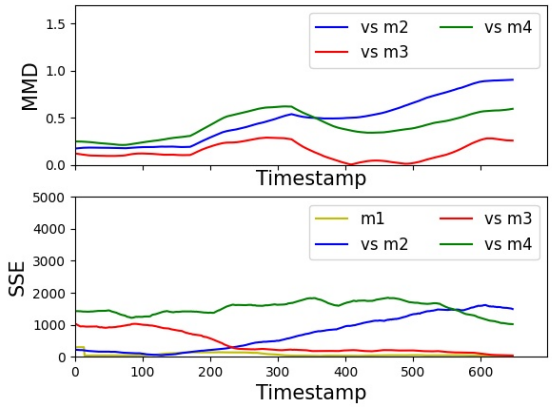
With model maintenance or replacement in MRM under data streams, SSE can be adopted to locally and efficiently monitor the entire knowledge reuse process on the borrower side (with $O(1)$ time and space complexity, as discussed in Section 3.4). Figure 3.19 shows that the sliding window length i does not significantly affect the ΔSSE evolution. The ΔSSE for m_3 achieves a stable state with variance $\text{var}(\Delta SSE) \to 0$ progressively.

Figure 3.19 shows that most ΔSSE values for m_2 are positive, indicating increasing SSE, while most ΔSSE values for m_3 are negative. Accumulation of positive ΔSSE values can degrade model reusability, triggering a new BLM process based on proactive forecasting.

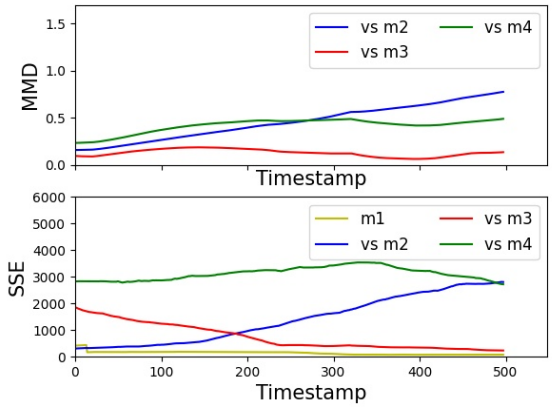
Analyzing $\Delta S_t = Z_t$, the trend level b_t is significant when $\zeta_t > 0$ since b_t determines the scale of Z_t. Trends and level coefficients b_t and ζ_t of the HW model are considered



(a)



(b)

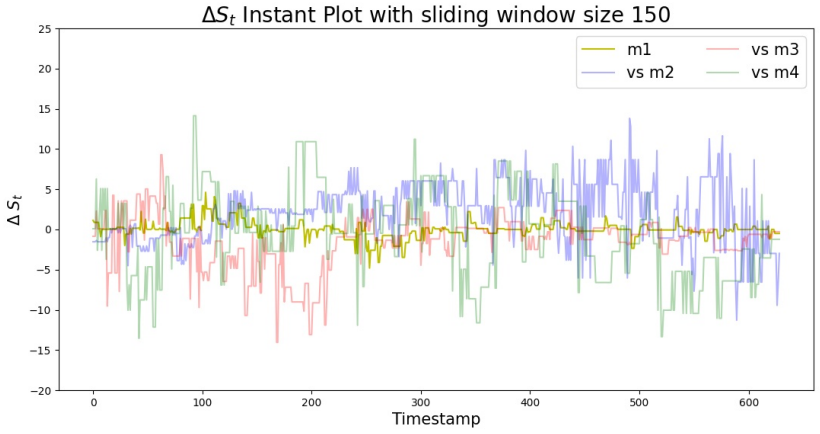


(c)

Figure 3.18: (a)(b)(c) MMD and SSE time series plots corresponding to different length i of sliding window, (a) $n_i = 100$, (b) $n_i = 150$, and (c) $n_i = 300$, in Scenario III.

constants when predicting Z_t. Figure 3.20 (a) shows that the trend effect $b_t^{(m_1)}$ lacks high peaks, indicating no significant future change for m_1. However, high peaks for $b_t^{(m_2)}$ and $b_t^{(m_4)}$ suggest that their forecasting may exceed acceptable ΔSSE expectations, triggering

(a)



(b)

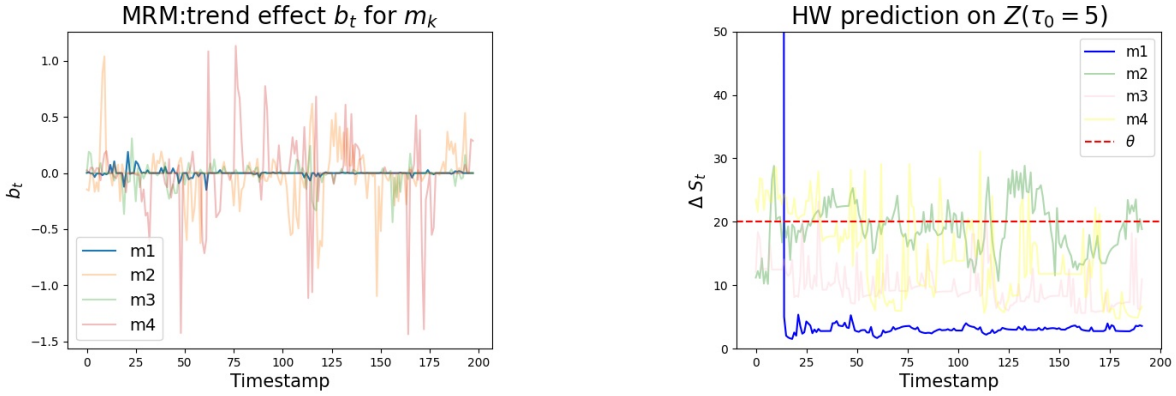
(c)

Figure 3.19: (a)(b)(c) Time series plots of ΔSSE corresponding to different length i of sliding window, (a) $n_i = 100$, (b) $n_i = 150$, and (c) $n_i = 300$, in Scenario III.

a new BLM process if the excess times surpass predefined tolerance levels.

Using Equation (3.16), we calculate $\hat{Z}_{t+\tau_0}$, estimating $Z_{t+\tau_0}$. Setting $\tau_0 = 5$ to balance

prediction accuracy and steps ahead, and choosing $\theta = 20$ as the threshold, Figure 3.20(b) shows $\hat{Z}^{(m_2)}_{t+\tau_0}$ and $\hat{Z}^{(m_4)}_{t+\tau_0}$ often exceed the threshold, indicating the need for a BLM process. The prediction interval is significantly affected by historical dataset variance.



(a) (b)

Figure 3.20: (a) Time series plot of b_t for HW model based on Z_t in Scenario III. (b) HW model: time series plot of the upper bound of the confidence interval for $\hat{Z}_{\tau_0=5}$ with $\theta = 20$ in Scenario III.

3.7 Limitations and Future Research

The paradigm of knowledge reuse in edge computing (EC) environments presents significant promise, but several limitations must be addressed to realize its full potential. Key challenges include system heterogeneity and the non-stationary nature of data streams.

System Heterogeneity: EC environments are characterized by a high degree of heterogeneity among devices, which vary significantly in computational power, energy capacity, and network connectivity. The current paradigm does not fully account for how these differences impact the performance and feasibility of knowledge reuse. Future research should focus on developing adaptive mechanisms that can dynamically adjust to the diverse capabilities and constraints of individual edge devices. This would ensure efficient and equitable utilization of resources across heterogeneous environments.

Non-stationary Data Streams: EC environments frequently handle dynamic and non-stationary data streams, which can lead to concept drifts that render reused models obsolete. Although the proposed Model Reusability Monitoring (MRM) mechanism provides some mitigation, there is a need for more sophisticated and proactive approaches to model adaptation. Future work should explore continuous learning and online adaptation techniques that can rapidly respond to changes in data distribution, thereby maintaining model accuracy over time.

Limitations in Handling Complex Data: The current method performs well

for linear regression and support vector regression, as the Maximum Mean Discrepancy (MMD) and Cosine Distance (CD) metrics are effective in capturing statistical patterns in relatively simple datasets. However, for more complex data types such as images or text, these methods may fall short due to their inability to extract hierarchical features. One potential solution is the integration of neural networks to extract meaningful features before applying MMD and CD, thereby extending the applicability of the method to complex data scenarios. Future work should investigate such hybrid approaches to overcome the constraints of model usability in diverse data contexts.

Addressing these limitations through targeted research will be crucial for advancing the knowledge reuse paradigm and enhancing the efficiency, scalability, and effectiveness of edge computing environments.

3.8 Conclusions

In this chapter, we introduce the knowledge reuse paradigm based on the premise that locally trained ML models on certain edge nodes can be reused by other nodes with similar datasets. We propose two dataset similarity algorithms based on minimum sufficient statistical knowledge derived from a modified kernel-driven maximum mean discrepancy metric and cosine dissimilarities of principal components. These synopses are exchanged among edge nodes to facilitate the knowledge reuse paradigm and provide a decision-making framework for feasible model reusability.

Our experimental evaluations over real and synthetic datasets demonstrate that the proposed algorithms require minimal information (statistical synopses) to be exchanged among nodes, resulting in a resource-aware lightweight BLM process. We showcase the advantage of reusing trained models over nodes with similar datasets through three experimental scenarios, eliminating the need to build models anew, thereby saving computational resources and communication overhead locally. Our algorithms ensure acceptable analytics quality via model reusability, as evidenced by the defined discrepancy performance metrics over widely adopted supervised and unsupervised learning models. This indicates the applicability of the knowledge reuse paradigm in resource-constrained environments, where borrower nodes can exploit and reuse models from loaners to support their predictive modeling and inferential analytics applications without expending resources on model retraining and verification.

In addition, given potential data changes in dynamic EC environments, we introduced a lightweight monitoring mechanism (MRM) over edge nodes that can autonomously assess whether reused models become obsolete or useless in terms of quality of analytics. In such cases, nodes can proactively forecast when to initiate the process of finding more appropriate models to be reused/adopted with high confidence. These results demonstrate

the practicality of deploying knowledge reuse mechanisms in real-world edge computing systems. By ensuring computational efficiency and scalability, the proposed approach supports the deployment of predictive models in resource-constrained environments while maintaining acceptable accuracy.

Therefore, we conclude that reusable models can be effectively identified using the proposed hypothesis test, and any violation of reusability can be detected by the monitoring system. This approach ensures that redundant model computation is avoided, assuming that the distributed data shares a *similar* distribution and that executable models on *loaners* exist. In Chapter 4, we will explore how to efficiently build a distributed learning paradigm in the absence of pre-trained models.

Chapter 4

Efficient Distributed Learning with Enhanced Reusability: A Multi-Task Learning Approach

This section relaxes the prevailing assumption of the model existing to reuse in Chapter 3, proposing a novel methodology for constructing models from scratch instead. This methodology leverages the concept of knowledge reuse in distributed computing environments, discussed in Section 2.4, to enhance both communication and computational efficiency. We assert that model reusability can be significantly enhanced through knowledge reuse, particularly by employing multi-task learning, thereby reducing the necessary computation for all clients.

4.1 Introduction

Recently, the field of distributed predictive modeling and analytics has garnered substantial interest, especially when these analytics are executed at the network edge, a concept commonly referred to as the Edge Computing paradigm. Simultaneously, there has been an emergence of distributed learning paradigms, which introduce several challenges. These challenges include task offloading, service migration (Kolomvatsos and Anagnostopoulos, 2022), and managing distributed data heterogeneity in devices that are constrained by their resources (Li et al., 2020c). The rapid expansion of computing nodes and the explosive growth in data volume in these environments necessitate analytics mechanisms that can substantially reduce computational and communication resource utilization. These mechanisms, particularly predictive and exploratory analysis, must leverage and reuse existing models and derived knowledge whenever feasible.

Section 3.2 discussed the application of the model reuse. However, these methods do not focus on enhancing the generalization capacity of reused models over complex data

structures, limiting their direct applicability to nodes with similar tasks. Additionally, some assume an ideal environment where reusable models can be applied directly by applications (Lee et al., 2019) without even training and adaptation to specific tasks. The challenges lie in differentiating between reusable and non-reusable models, significantly impacting the quality of analytics, such as prediction accuracy and generalization capacity; and how to improve the local model performance through collaborative training.

Multitask Learning (MtL) is a machine learning paradigm that aims to improve generalization by learning multiple related tasks simultaneously, rather than independently. The core idea behind MtL is that certain tasks share commonalities or statistical properties, which can be leveraged to improve the performance of all tasks involved. This is achieved by optimizing multiple loss functions together within a unified framework, enabling shared learning across tasks. By doing so, MtL encourages the model to learn generalized features that are beneficial for all tasks, rather than overfitting to specific task datasets.

MtL can be categorized into two primary variants based on the nature of the tasks. Homogeneous Multitask Learning deals with tasks of the same type or family, such as multiple classification or regression tasks (Zhang and Yang, 2021). In this setting, tasks are expected to have consistent input and output types, and the shared learning process is often more straightforward. Heterogeneous Multitask Learning focuses on tasks that are fundamentally different in nature, such as a combination of classification, regression, or clustering tasks (Li et al., 2014). Heterogeneous MtL introduces additional complexity due to the diverse requirements of each task type and is beyond the scope of this thesis.

In this thesis, we focus on homogeneous MtL to tailor reusable models in distributed computing environments, as evidenced by our experimental evaluation. Instead of training independent models locally on nodes with limited generalizability, the MtL paradigm enables collaboratively trained models that can learn and complete tasks across all nodes. These tailored models leverage the data from all nodes, allowing them to generalize effectively by capturing shared patterns and representations across tasks.

While data redundancy among similar edge devices is a known issue, it is important to recognize the limitations of homogeneous data in improving generalization. Training on data that is too similar can lead to overfitting and limit the model's adaptability to unseen tasks or nodes. To address this, incorporating learning from devices with relatively dissimilar data can help mitigate overfitting and foster more robust performance. This introduces a key challenge in the MtL framework: determining *which tasks should be jointly learned in a distributed way to generate reusable models with improved generalization capacity.*

By addressing this challenge, MtL offers a promising approach for building efficient, scalable, and reusable models in distributed environments, making it a cornerstone of modern distributed machine learning research.

To address the above challenges, we develop an innovative two-phase Distributed Multitask Learning (DMtL) framework involving clustering and dissemination. The framework consists of the following components:

- Clustering Phase: This phase uses Partial Learning Curves (PLC) to assess and group similar tasks from different nodes based on their performance metrics.
- Dissemination Phase: In this phase, selected head nodes carry out distributed multitask learning to improve and finalize model training.

4.2 Related Work

Multitask Learning (MtL) is employed when learning over multiple tasks simultaneously, with its fundamental component being the learning of task similarities (Zhang and Yang, 2021; Pan and Yang, 2009). The method in Thrun and O'Sullivan (1998) grouped tasks into classes of mutually related processing activities using the ϵ-optimal distance metric to determine which tasks should be involved in learning. The multitask clustering presented in Cao et al. (2018) exploited correlations among task features based on the similarity between relative means over tasks, assuming that data follow similar distributions. Additionally, the method proposed in Shui et al. (2019) learned task relationships using the Wasserstein distance between data distributions.

Recent works have demonstrated the efficiency of MtL in a centralized mode, such as in cloud data centers. A perceptron-based centralized approach for MtL has been proposed in Cavallanti et al. (2010), introducing the *task similarity/relationship matrix* among different data distributions, where each entry represents the similarity between tasks. Hereinafter, we denote the task relationship matrix by $\mathbf{\Omega}^{-1}$ for notation compatibility with related work. In Cavallanti et al. (2010), $\mathbf{\Omega}^{-1}$ is assumed to be known in advance, an assumption that does not hold in distributed computing environments where tasks and data are not entirely or even partially shared or known in advance. The possibility of having exactly the same task-relatedness is rare. Unlike Cavallanti et al. (2010), we contribute to the field with a *distributed learning* approach that learns the task relationship matrix entries via local training of the nodes' models based on Learning Curves (LCs). Our method achieves task relationship matrix learning before the DMtL process, relaxing the assumption of known task relationships and becoming practical as this information is extracted directly from nodes' data rather than being set manually or known in advance.

The approaches in (Wang et al., 2016; Liu et al., 2017; Smith et al., 2017) achieved task relationship learning using regularized objective functions for MtL in a distributed manner. Wang et al. (2016) referred to a communication-efficient Lasso-based DMtL comparable to its centralized counterpart. The framework in Liu et al. (2017), DMTRL, introduced a

dual form of the general MtL problem. In the same context, Federated Learning is adopted to reduce communication costs in DMtL. The FL-MtL framework in Smith et al. (2017) (MOCHA) solved general MtL problems by updating the models' weights (parameters), hereinafter denoted as $\mathbf{W}$ for notation compatibility. Both DMTRL and MOCHA updated the task relationship matrix $\mathbf{\Omega}^{-1}$ after updating the model parameters matrix $\mathbf{W}$ under the assumption that $\text{tr}(\mathbf{\Omega}^{-1}) = 1$. Here, $\text{tr}(\mathbf{A})$ denotes the trace of a square matrix $\mathbf{A}$, defined as the sum of elements on its main diagonal. In our approach, this assumption does not hold true as $\mathbf{\Omega}^{-1}$ arbitrarily reflects information captured by the models' LCs to estimate task relationships.

The MtL approach in (Li et al., 2019a) addressed the challenge of new nodes joining the system, incrementally updating and extending the matrices $\mathbf{\Omega}^{-1}$ and $\mathbf{W}$ for newly incoming nodes. Compared with centralized MtL approaches, which update and aggregate model parameters in a centralized location (e.g., cloud), our PLC-based DMtL method spans across only *selected* nodes in a decentralized manner. Our task relationship matrix learning relies on communication only among selected nodes rather than exchanging model weights among all nodes without requiring a centralized location. In our method, communication costs are significantly reduced, as cloud/data servers are traditionally far from distributed nodes (e.g., Edge Computing), while short-distance routes ensure less communication cost in decentralized learning.

4.3 Methodology

4.3.1 Preliminaries

Table 2.2 and 2.3 summarize the notation used in this chapter. The hypothesis-related problem and the fundamentals of the learning paradigm are explained as follows.

Hypothesis 3 (Data Sufficiency)**.** *Assuming sufficient data availability for training local models, significant improvements in model generalization performance are not expected when the data distributions of the two nodes are similar.*

Considering a specific scenario stated in Hypothesis 3, the challenge can be defined as follows:

Problem 3 (Model Reuse Optimization (MRO))**.** *Considering that models are not trained, is it possible to train them from scratch using the concept of knowledge reuse? While the models are selected to be reused, the challenge is to enhance the borrowed model's predictive performance without increasing computation and communication costs.*

Without executed results/models, an efficient approach is to select as few nodes as possible to train the models without compromising performance. The proposed method

focuses on determining which intermediate results are sufficient for reuse, enabling only a subset of nodes to complete the full model training process.

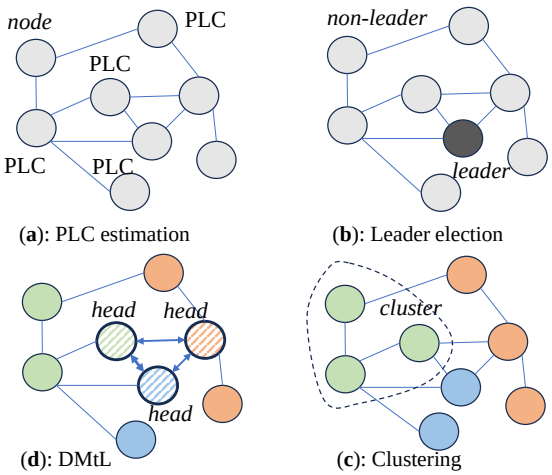


Figure 4.1: Illustration of the stages in the Distributed Multi-task Machine Learning (DMtL) framework: (a) PLC estimation shows nodes estimating Partial Learning Curves to assess model performance; (b) Leader election where nodes select a leader based on predefined criteria; (c) Clustering where similar nodes are grouped based on their PLC values; (d) DMtL execution where designated head nodes coordinate task distribution within clusters.

The paradigm addresses situations where no pre-trained models are available or where the existing models may not be entirely applicable to new tasks, i.e., Problem 3. Specifically, we introduce a two-phase Distributed Multi-task Machine Learning (DMtL) framework designed to address this challenge effectively. Similar tasks are identified and grouped in the **Clustering Phase** based on performance meta-features extracted from locally trained models. This clustering leverages Partial Learning Curves (PLC) to assess and categorize task similarity efficiently. In the subsequent phase, **Dissemination Phase**, we utilize the insights gained from the PLC-driven analysis within the DMtL paradigm to enhance the performance of candidate reusable models. This approach is specifically tailored for distributed computing environments, where it significantly boosts the efficiency and efficacy of task-specific model reuse. The details are depicted in Figure 4.1, which will be further explained in the following.

Distributed Multitask Learning:

Consider a distributed system consisting of M nodes denoted by $\mathcal{M} = \{1, 2, \ldots, M\}$.

These nodes are interconnected based on a network topology described by a directed graph $\mathcal{G}(\mathcal{M}, \mathcal{E})$. The connections between nodes are captured by the adjacency matrix $\mathcal{E} = [e_{i,j}] \in \mathbb{R}^{M \times M}$, which has specific null space and spectral properties. The neighborhood of a node i, represented as $\mathcal{M}_i = \{j \in \mathcal{M} : e_{i,j} > 0\}$, consists of all nodes j that can directly communicate with node i. If $e_{i,j} = 0$, it indicates that there is no communication link between nodes i and j, meaning $j \notin \mathcal{M}_i$. Note that $e_{i,j} \neq e_{j,i}$ may occur for $i \neq j$, reflecting asymmetrical communication. The adjacency matrix $\mathcal{E}$ remains constant throughout the distributed machine learning (DMtL) process. Each node i acquires data $\{\mathbf{X}_i, \mathbf{Y}_i\} \sim \mathcal{P}_i$ from its local environment, where $\mathbf{X}_i \in \mathbb{R}^{n_i \times d}$ represents the input features and $\mathbf{Y}_i \in \mathbb{R}^{n_i}$ represents the corresponding outputs, sampled from a joint probability distribution $\mathcal{P}_i$. For any two distinct nodes i and j (i.e., $i \neq j$), their joint probability distributions $\mathcal{P}_i$ and $\mathcal{P}_j$ can either be similar ($\mathcal{P}_j \approx \mathcal{P}_i$) or different ($\mathcal{P}_j \neq \mathcal{P}_i$), indicating that the data distributions across nodes can vary.We present a Multi-task Learning (MtL) framework that is formulated on a regularization-based approach, enabling effective learning across distributed nodes as in Problem 4.

Problem 4. *Given a local dataset for each node $i \in \mathcal{M}$, denoted as $\mathbf{X}_i \times \mathbf{Y}_i \in \mathbb{R}^{n_i \times (d+1)}$, the objective is to determine model weights $\boldsymbol{\omega}_i \in \mathbb{R}^d$ for all M nodes and optimize the task relationship matrix $\boldsymbol{\Omega}^{-1}$. The goal is to minimize the following regularized objective, incorporating various convex loss functions $\mathcal{L}_i$, for $i = 1, \ldots, m$:*

$$\min_{\mathbf{W}, \boldsymbol{\Omega}} \Big\{ \sum_{i=1}^{n} \sum_{t=1}^{n_i} \mathcal{L}_i(\boldsymbol{\omega}_i^\top, (x_i^t, y_i^t)) + \frac{\lambda_1}{2} tr(\mathbf{W} \boldsymbol{\Omega}^{-1} \mathbf{W}^T) + \frac{\lambda_2}{2} \|\mathbf{W}\|_F^2 \Big\} \tag{4.1}$$

with $\lambda_1, \lambda_2 > 0$ as regularization parameters controlling the complexity of $\mathbf{W}$ and $\boldsymbol{\Omega}^{-1}$. When $\boldsymbol{\Omega}^{-1} = \mathbf{I}_{i \times m}$ and $\lambda_1 + \lambda_2 = \lambda$, the multi-task learning framework reduces to independent single task learning, involving only the squared Frobenius norm of $\mathbf{W}$.

In distributed learning environments, the Equation (4.1) is decentralized and solved collaboratively across nodes. This setup allows for the exchange of model parameters rather than raw data, preserving data privacy and improving training performance regardless of whether data across nodes are non-i.i.d. While tasks at different nodes may differ, collaborative learning can reveal latent information, enhancing overall model generalization. If Hypothesis 3, discussed in Section 4.3.1, holds, our Distributed Multi-task Learning (DMtL) framework strategically can involve only the head nodes (selected as K out of M) during the learning process, thus excluding redundant or similar tasks to optimize computational resources and learning efficiency. Note that this approach aligns with the decentralized nature of distributed computing, where model parameters are exchanged to enhance local computations without central data aggregation.

Partial Learning Curves:

Learning Curves (LCs) are pivotal for comparing model performance across experiences, particularly for evaluating how models generalize from historical data to new, unseen datasets. Following Leite and Brazdil (2005), we utilize LCs to predict model performance on new data by analyzing trends over historical data. Our framework extends this concept to assess the similarity between nodes' local models, facilitating their grouping based on shared performance characteristics.

LCs not only depend on the training data but also on the model configuration (Viering and Loog, 2022), capturing the model's learning behavior through generalization performance against training examples. Consider a local Support Vector Machine (SVM)-based classifier on node i, defined as:

$$f_i : \text{sign}(\hat{\mathbf{W}}_i^\top \Phi(\mathbf{X}_i^{train}) + \mathbf{b}_i) \rightarrow \hat{\mathbf{Y}}_i, \tag{4.2}$$

where f_i maps input $\mathbf{X}_i$ to output $\mathbf{Y}_i$, $\hat{\mathbf{W}}_i$ represents the normal to the hyperplane, and $\Phi(\mathbf{X}_i)$ denotes the transformed data space mapping.

The classification error, measured by cross-entropy loss, for this model is:

$$\mathcal{L}_i = -\frac{1}{n_i^T} \sum_{l=1}^{n_i^T} \left[y_l^T \log(\hat{y}_l^T) + (1 - y_l^T) \log(1 - \hat{y}_l^T) \right], \tag{4.3}$$

where $n_i^T = n_i - n_i^{Test}$ denotes the training set size. This leads to an expected generalization error $\mathbb{E}_{\mathcal{P}_i}[\mathcal{L}_i]$, indicative of the model's ability to predict new data.

4.3.2 Initial Stages

The initial stages (a) and (b) depicted in Figure 4.1 outline the preprocessing steps essential to our framework.

PLC Computation

Stage (a) involves the computation of Partial Learning Curves (PLCs). We define a Partial Learning Curve (PLC) that summarizes model performance over a spectrum of training set sizes, denoted by an index set $\mathcal{S}$. Each PLC element is computed as:

$$\hat{V}_i^{(S_s)} = \frac{1}{L} \sum_{\ell=1}^{L} \mathcal{L}_i^{(S_s)(\ell)}, \tag{4.4}$$

where $\hat{\mathbf{V}}_i = [\hat{V}_i^{(S_1)}, \ldots, \hat{V}_i^{(S_p)}]^\top$ represents the PLC vector in $\mathcal{R}^p$, providing a comprehensive measure of model reliability and generalization capability across different training

conditions. Then the similarity score is to compute distance calculation between $\hat{\mathbf{V}}_i$ and $\hat{\mathbf{V}}_j$ with devices i and j are connected.

The example of partial learning curves (PLCs) for training an SVM on five nodes is illustrated in Figure 4.2. The data for the five nodes are simulated, with nodes 1 and 2 sharing a more similar distribution. From Figure 4.2, it is evident that the PLCs for nodes 1 and 2 are closer to each other than the others, highlighting and verifying the clustering efficiency of PLCs. The details of PLC computation are as follows.

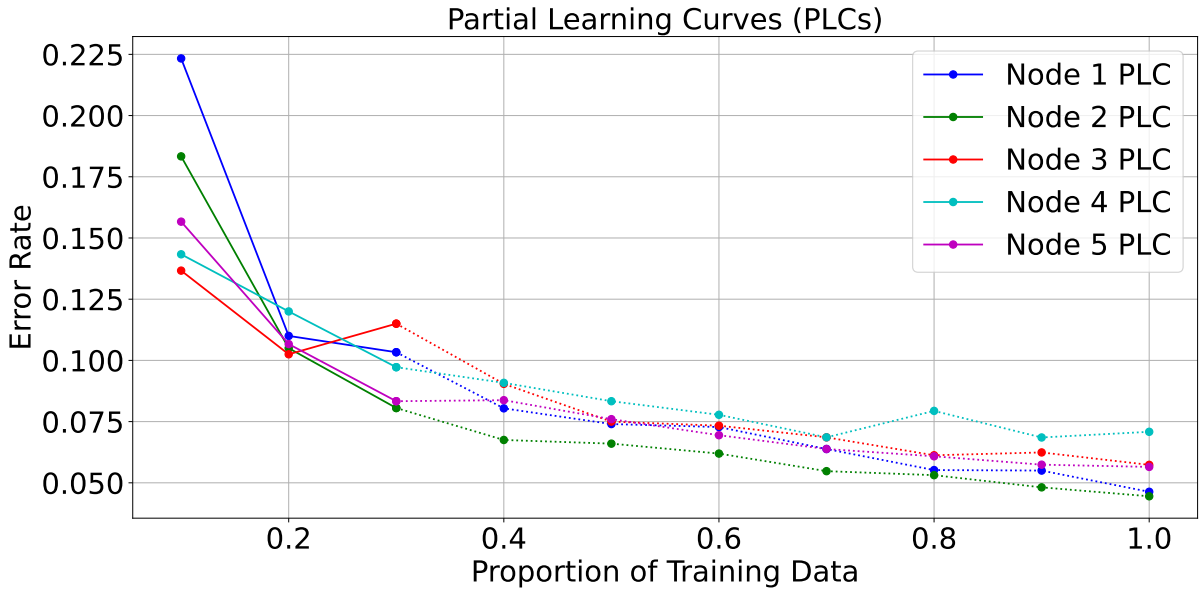


Figure 4.2: Example of Partial Learning Curves (PLCs) for different nodes. Solid lines represent $\frac{s}{n_i} \leq 30\%$, and the dotted lines indicate the incomplete parts for learning curve prediction.

Considering the variability of loss functions and the need for practical error estimation methods, we employ bootstrapping (Efron, 1983) to approximate the LC for a local model. Given the dataset $\mathbf{X}_i \in \mathbb{R}^{n_i \times d}$, bootstrapping involves:

1. Randomly resampling the data with replacement to construct several subsets $\{\mathbf{X}_i^{(s)(\ell)}\}$ of size $s \leq n_i$.

2. Training the model on each subset and computing the training error $\mathcal{L}_i^{(s)(\ell)}$.

3. Averaging these errors to estimate the generalization performance.

The procedure of PLC computation is shown in Algorithm 5

Leader Election

Stage (b) in Figure 4.1 concerns the election of a leader node, which is typically selected based on criteria such as device capabilities or statistical meta-features extracted during

Algorithm 5 Bootstrapped PLC estimation on node i

Input: Node i with data $\mathbf{X}_i$; training sample sizes set $\mathcal{S}$; bootstrapping L.
Output: PLC estimated vector $\hat{\mathbf{V}}_i$

```
for each S_s ∈ S do
    for each round ℓ = 1 ... L do
        X_i^{(S_s)(ℓ)} ← Resample X_i w. replacement in size S_s
        Compute training error L_i^{(S_s)(ℓ)} on X_i^{(S_s)(ℓ)}
    end for
    Compute V̂_i^{(S_s)} using (4.4)
end for
V̂_i ← Rearrange V̂_i^{(S_s)} orderly using (4.4)
```

the PLC calculation. Specifically, after nodes exchange their $\hat{v}_i$ values to all nodes, the node exhibiting the highest generalization error, indicated by values of $\hat{v}_i$, is chosen as the leader.

4.3.3 Learning Paradigm

Clustering Phase

While a leader is selected, we introduce a distributed learning paradigm where a leader node coordinates the learning process across non-leader nodes by leveraging their Partial Learning Curves (PLCs). The leader node computes the task relationship matrix $\mathbf{\Omega}^{-1}$ entries based on the PLC-derived distances between nodes.

$$\mathbf{\Omega}^{-1}_{i,j} = \frac{2}{M} \cdot \frac{1}{1 + \exp(\epsilon \cdot d_{i,j})}, \tag{4.5}$$

where $d_{i,j} = \|\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j\|_2$ is the Euclidean distance between nodes' PLC vectors, and $\epsilon > 0$ is a tunable parameter to scale the sensitivity of the interaction.Note that Equation (4.5) captures the similarity between the predictive performance of models trained on different nodes, making it well-suited for identifying related tasks in distributed multitask learning environments. By leveraging the PLCs, this method effectively presents the inherent similarities in tasks, which are crucial for optimizing distributed learning and enhancing model generalizability.

The leader then clusters nodes into K groups using a Gaussian Mixture Model (GMM), where each node i with its PLC $\hat{\mathbf{V}}_i$ is probabilistically assigned to a cluster $\mathcal{C}_k$ based on the highest posterior probability:

$$P(\mathcal{C}_k|\hat{\mathbf{V}}_i) = \max_{\kappa=1,\dots,K} \{P(\mathcal{C}_\kappa|\hat{\mathbf{V}}_i)\} \tag{4.6}$$

$$P(\mathcal{C}_k|\hat{\mathbf{V}}_i) = \frac{\phi_k \mathcal{G}(\hat{\mathbf{V}}_i|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{\kappa=1}^{K} \phi_\kappa \mathcal{G}(\hat{\mathbf{V}}_i|\boldsymbol{\mu}_\kappa, \Sigma_\kappa)} \tag{4.7}$$

The clustering parameters, including the mixture weights $\{\phi_\kappa\}$, mean vectors $\{\boldsymbol{\mu}_\kappa\}$, and covariance matrices $\{\Sigma_\kappa\}$, are optimized using the Expectation-Maximization algorithm. The clustering results are utilized to designate head nodes for each cluster, facilitating the DMtL process as outlined in DP. The nodes' PLC-based relationships and task alignments are represented in the $\boldsymbol{\Omega}_{K\times K}$ matrix:

$$\boldsymbol{\Omega}^{-1}_{K\times K}(i,j) = \boldsymbol{\Omega}^{-1}_{M\times M}(i,j), \quad i \in \mathcal{C}_k, \quad j \in \mathcal{C}_\kappa \tag{4.8}$$

ensuring that each head node receives all necessary parameters to synchronize the learning activities across the distributed network effectively. The details of CP are summarized in Algorithm 6.

Algorithm 6 The PLC-based Clustering Phase

Input: M nodes; number of clusters K; PLC dimension p
Output: K clusters with heads; cluster-based matrix $\boldsymbol{\Omega}^{-1}_{K\times K}$

1: **for** each node $i \in \mathcal{M}$ **do**
2: Send PLC $\hat{\mathbf{V}}_i$ in (4.4) to leader.
3: **end for**
4: /***Leader Side***/
5: Calculate matrix $\boldsymbol{\Omega}^{-1}_{M\times M}$ in (4.5)
6: Group nodes using GMM in K clusters over $\{\hat{\mathbf{V}}_i\}$.
7: Assign K heads to clusters using (4.6).
8: Send matrix $\boldsymbol{\Omega}^{-1}_{K\times K}$ in (4.8) to heads.

Dissemination Phase

In DP, we propose a PLC-based DMtL process based on decentralized SGD optimization to solve Problem 3 in a distributed manner. Our DMtL engages only the heads obtained by Algorithm 7 to solve the *distributed version* of Equation (4.1). Unlike Smith et al. (2017), Zhang and Yeung (2010), and Li et al. (2019a), which iteratively update the task relationship matrix $\boldsymbol{\Omega}^{-1}$, we construct this matrix using existing knowledge of the performances of local models (PLCs).

Given that $K \ll M$, the cluster-based task relationship matrix $\boldsymbol{\Omega}^{-1}_{K\times K}$ obtained from the CP is a *compressed* version of the entire matrix $\boldsymbol{\Omega}^{-1}_{M\times M}$. We formulate our DMtL problem across K heads as follows:

Given K heads from clusters $\{\mathcal{C}_k\}_{k=1}^{K}$, their corresponding datasets $\mathbf{Z}_k = \{\mathbf{X}_k, \mathbf{Y}_k\}$ of sizes n_k, and the cluster-based task relationship matrix $\boldsymbol{\Omega}^{-1}_{K\times K}$, we seek the model weight

matrix $\mathbf{W} = [\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_K] \in \mathbb{R}^{d \times K}$ that minimizes the regularized objective:

$$\mathcal{J}(\mathbf{W}) \quad = \quad \min_{\mathbf{W}} \left\{ \sum_{k=1}^{K} \sum_{t=1}^{n_k} \mathcal{L}_k(\boldsymbol{\omega}_k^\top, (\mathbf{x}_k^t, y_k^t)) \; + \; \frac{\lambda_1}{2} \operatorname{tr}(\mathbf{W}\boldsymbol{\Omega}_{K\times K}^{-1}\mathbf{W}^\top) + \frac{\lambda_2}{2}\|\mathbf{W}\|_F^2 \right\} \quad (4.9)$$

In Equation (4.9), the heads *do not exchange any data* during learning. The optimal model weights in $\mathbf{W}$ correspond to the *tailored models* f_k^* of the heads, which can be reused by cluster members for future analytics tasks. These members have not been engaged in this learning process.

Each tailored model f_k^* captures the generalization characteristics of both the local models of members in cluster $\mathcal{C}_k$ via their representative head and the models outside of that cluster, i.e., the local models from nodes in external clusters $\mathcal{C}_\kappa$ with $\kappa \neq k$. This significantly differs from other MtL methodologies, which do not consider the PLC model performances for model re-usability and generalization. For instance, the approach in Liu et al. (2017) updates the task relationship matrix by assuming $\boldsymbol{\Omega}$ is fixed as $\frac{1}{K}\mathbf{I}_{K\times K}$. The solution is $\boldsymbol{\Omega}^{-1} = \frac{(\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}}{\operatorname{tr}((\mathbf{W}^T\mathbf{W})^{\frac{1}{2}})}$ with the constraint $\operatorname{tr}(\boldsymbol{\Omega}^{-1}) = 1$, reducing from MtL to Single Task Learning (StL) without exploiting local model (PLC) information for enhancing the generalizability of trained models.

Our DMtL initializes the model weight matrix $\mathbf{W}$ with the initially trained models $\{\boldsymbol{\omega}_k\}_{k=1}^K$ of the head nodes from the CP at round $t = 0$, i.e., $\mathbf{W}^{t=0} = [\boldsymbol{\omega}_1^{t=0}, \ldots, \boldsymbol{\omega}_K^{t=0}]$. Each head model $\boldsymbol{\omega}_k$, $k = 1, \ldots, K$, is the optimal existing model with error tolerance γ_p. Thus, we decrease training rounds and save storage *without* tuning the cluster-based task relationship matrix $\boldsymbol{\Omega}$.

To engage the K head nodes in distributed training, we split Equation (4.9) into local quantities, one per head node. The local updates in each head node yield the global minimization of (4.9).

Without loss of generality, we experiment with binary SVM classification adopting the hinge loss. The gradient of (4.9) with respect to the model weight of the head node is given by:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\omega}_k^{(t)}} = -\sum_{t=1}^{n_k} \mathbf{1}[y_k^{(t)} \boldsymbol{\omega}_k^\top \mathbf{x}_k^{(t)} < 1] y_k^{(t)} \mathbf{x}_k^{(t)} + \lambda_1 (\mathbf{W}\boldsymbol{\Omega}_{K\times K}^{-1})_k + \lambda_2 \boldsymbol{\omega}_k. \quad (4.10)$$

The second term in (4.10) allows the head node of $\mathcal{C}_k$ to acquire information from its peer head nodes outside its cluster, supporting model re-usability. The term $(\mathbf{W}\boldsymbol{\Omega}_{K\times K}^{-1})_k$ in (4.10) refers to the k^{th} column of the product matrix, a $\mathbb{R}^{d\times 1}$ weight vector. Rewriting

the product matrix as a summation of products, we obtain:

$$\boldsymbol{\omega}_k \boldsymbol{\Omega}_{kk}^{-1} + \sum_{\kappa \neq k}^{K} \boldsymbol{\omega}_\kappa \boldsymbol{\Omega}_{k\kappa}^{-1}. \tag{4.11}$$

This indicates that the model weight for the head node of $\mathcal{C}_k$ is updated with the sum of the model weights of head nodes outside $\mathcal{C}_k$, weighted by the task relationship matrix entries (PLC-wise similarity quantities).

At round t, each head of clusters $\mathcal{C}_k$ shares the model weight $\boldsymbol{\omega}_k^{(t)}$ with its peer heads. After receiving model weights from its peers $\{\boldsymbol{\omega}_\kappa\}, \kappa \in \{1, \ldots, K\} \setminus \{k\}$, the head node updates its model weight $\boldsymbol{\omega}_k^{(t)}$ using (4.10) as:

$$\boldsymbol{\omega}_k^{(t)} = \boldsymbol{\omega}_k^{(t-1)} - \eta \cdot \frac{\partial \mathcal{J}}{\partial \boldsymbol{\omega}_k^{(t-1)}} \tag{4.12}$$

where $\eta \in (0, 1)$ is the learning rate. The details of the PLC-based DMtL process are provided in Algorithm 7.

Algorithm 7 The PLC-based DMtL Process using $\boldsymbol{\Omega}_{K\times K}^{-1}$

Require: K heads; compressed matrix $\boldsymbol{\Omega}_{K\times K}^{-1}$
Ensure: DMtL-tailored models $\{f_k^*\}_{k=1}^K$
1: Each head initializes its model weight $\boldsymbol{\omega}_k^{t=0}$ from CP.
2: **for** round $t = 1, \ldots, T$ at head k **do**
3: Share $\boldsymbol{\omega}_k^{(t)}$ among K heads;
4: Aggregate and update $\boldsymbol{\omega}_k^{(t)}$ using (4.11), (4.10), and (4.12)
5: **end for**
6: Each head of cluster $\mathcal{C}_k$ returns its tailored model f_k^*.

Any analytics task, such as regression or multi-class classification (the latter easily transformed to multiple binary problems adopting a *one-vs-rest* strategy), and various loss functions, such as square loss, can be used for Equation (4.9).

4.4 Theoretical Analysis

In this section, we provide a rigorous theoretical analysis of our proposed DMtL framework. We first demonstrate the convergence properties of the decentralized SGD optimization process for solving the multi-task learning objective. We then extend our analysis to the convergence of SVM model parameters using mini-batch SGD, establishing the conditions under which our approach ensures reusable model parameters.

Lemma 1. *The training process for solving Equation* (4.9) *using the objective* $\mathcal{J}(\mathbf{W})$ *in (4.9) converges with SGD.*

Proof. See Appendix A.2.2 □

Lemma 2. *The training process for SVM model parameters using mini-batch SGD converges to a set of reusable parameters $\boldsymbol{\omega}_{i+1}$ following the update rule:*

$$\boldsymbol{\omega}_{i+1} = (1 - \eta_i \lambda)\boldsymbol{\omega}_i + \frac{\eta_i}{|\mathbf{Z}_i^*|} \sum_{j \in \mathbf{Z}_i^*} y_j \mathbf{x}_j, \tag{4.13}$$

where $\mathbf{Z}_i^* = \{j \in \mathbf{Z}_i : y_j \boldsymbol{\omega}_i^T \mathbf{x}_j < 1\}$.

Proof. See Appendix A.2.2 □

4.5 Experimental Setup

This section introduces the dataset, performance metrics, and baseline used for experiments. In the synthetic dataset (SD), we assess the performance of PLC-based clustering for model reusability. For the CIFAR-10 and Sentiment datasets, the heads are formed based on Algorithm 6. We investigate whether the PLC-based DMtL models are comparable with other multi-task learning (MtL) and federated learning (FL) approaches. In the Sentiment and CIFAR-10 datasets, our DMtL employs $K = 4$ and $K = 5$ clusters, respectively. The value of K in all the experiments is determined through hyperparameter tuning (see Section 4.5.3). Each experiment is run 100 times per case using GeForce RTX 3090 GPUs. Note that the use of a desktop GPU (RTX 3090) in our simulations serves as a proxy to evaluate the feasibility and performance of the proposed methods in edge computing (EC) scenarios. While real-world EC nodes have limited computational resources compared to high-end GPUs, the simulations provide an environment where the algorithmic behaviors and interactions can be rigorously tested under controlled conditions.

4.5.1 Datasets

Synthetic Dataset

The binary classification local models were trained on a synthetic dataset (SD) within scikit-learn package [1] for clustering, consisting of 2×10^5 data points in $d = 20$ dimensions. The class labels $y \in \{0, 1\}$ were assigned to data points with probabilities p' and $1 - p'$, respectively.

We define a step size $a > 0$ to evenly divide the interval [0, 1] and obtain different configurations of $\{p', 1-p'\}$. For example, with $a = 0.2$, we obtain assignment probabilities

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs`

$p' \in \{0.2, 0.4, 0.6, 0.8\}$ and $1 - p'$ for labeling data as 0 and 1, respectively, resulting in class imbalanced cases. Each task is assigned a different p' randomly.

We re-sampled 10^4 points from the SD to create $M \in \{100, 1000\}$ local datasets distributed across $M \in \{100, 1000\}$ nodes. The number of clusters was set to $K = 8$ and $K = 22$ for $M = 100$ and $M = 1000$, respectively.

Real and Benchmark Datasets

For image classification tasks, we use the CIFAR-10[2] dataset. CIFAR-10 consists of 6×10^4 color images (each image is 32×32 pixels), each assigned to one of the 10 class labels $y \in \{0, \ldots, 9\}$, with 6000 images per class. Each node contains 3000 images, distributed across $M = 20$ nodes. For the SGD-based SVM classifier model, we adopted the VGG16 Convolutional Neural Network (CNN) (Simonyan and Zisserman, 2015) for feature extraction, resulting in $d = 512$ features per image.

We also experimented with the Sentiment dataset[3], which consists of four sub-datasets corresponding to products: books, electronics, kitchens, and DVDs from Amazon.com. Each dataset contains 1000 positive (0) and 1000 negative (1) observations with different features. We summarized the word frequencies and obtained the total number of unique words (or short sentences) for each category as follows: 196,598 for books, 112,126 for electronics, 94,191 for kitchens, and 189,502 for DVDs. We unified all the features to build a Bag-of-Words model (Ko, 2012) for all data, resulting in a total feature dimension of $d = 473,856$. Based on the overall frequency of these features, we selected those with a frequency greater than 36, referred to as *pivot features*. The final size of each (complete) dataset is $\mathbb{R}^{2000 \times 5035}$ ($d = 5035$). The number of nodes for Sentiment data is $M = 40$.

4.5.2 Performance Metrics

We choose the classification accuracy $\alpha_i \in [0, 1]$ of a classifier f_i trained on $\mathbf{X}_i$ of node i as a comparison metric with the baseline and comparison models . We introduce the *model reuse performance* metrics of f_i reused for classification over other nodes' tasks. For each pair of nodes (i, j) with $i \neq j$, $\alpha_{ij} \in [0, 1]$ is the classification accuracy of f_i, which is trained on node i over $\mathbf{X}_i$ and used by node j being tested on $\mathbf{X}_j$. If the data of nodes in a cluster are i.i.d., we expect α_{ij} to be the same as the generalization performance of f_i built over $\mathbf{X}_i$.

We define $\xi \in \mathbb{R}$ as the average *difference in reusable model accuracy* between two factors when we randomly select a pair of nodes for model reuse: $\xi = \mu_{out} - \mu_{in} =$

[2]Publicly available at: `https://www.cs.toronto.edu/~kriz/cifar.html`
[3]Publicly available at: `https://www.cs.jhu.edu/~mdredze/datasets/sentiment/`

$\frac{1}{K}\sum_{k=1}^{K}\xi_k$ with

$$\xi_k = \mu_{out}^{(k)} - \mu_{in}^{(k)} = \frac{1}{M_k(M-M_k)}\sum_{i=1}^{M_k}\sum_{l=1}^{M-M_k}(\alpha_l - \alpha_{il}) - \frac{1}{M_k^2}\sum_{i=1}^{M_k}\sum_{j=1}^{M_k}(\alpha_j - \alpha_{ij}), \quad (4.14)$$

where index l runs over all the nodes which do not belong to cluster $\mathcal{C}_k$, $k = 1, \ldots, K$. μ_{out} indicates the average model reuse accuracy degradation *outside* each cluster. It measures the average performance drop of a reusable model when used by nodes outside its originating cluster. μ_{in} is the average model reuse performance degradation *within* each cluster, indicating the average prediction performance drop of a model used by any other node in the same cluster.

The $\mu_{in}^{(k)}$ and $\mu_{out}^{(k)}$ in (4.14) represent model reuse performance degradation inside and outside cluster $\mathcal{C}_k$, respectively. We denote with μ_{in}^* and μ_{out}^* the versions where we select the heads' tailored models $\{f_k^*\}_{k=1}^K$ only for reuse, corresponding to μ_{in} and μ_{out}, respectively, and obtain ξ^*.

Note: (i) The smaller the μ_{in}, the better the model reuse performance due to the reusability of models from nodes within clusters. (ii) A small μ_{out} indicates the effectiveness and applicability of our DMtL in offering reusable models to nodes outside the cluster that provides these models. (iii) A positive and small $\xi > 0$ indicates that reusing models within clusters suffers less performance degradation compared to reusing models by nodes outside clusters. When all nodes share similar i.i.d. data, we expect $\xi \approx 0$ due to $\mu_{in} \approx \mu_{out} \approx 0$.

We adopt the Sørensen-Dice coefficient Dice (1945), μ_{DC}, to evaluate the PLC-driven clustering efficiency, as it measures the similarity between datasets. A $\mu_{DC} = 0$ between datasets suggests zero similarity; $\mu_{DC} = 1$ indicates two identical datasets. μ_{DC} reflects the similarity among tasks of nodes within the same cluster and the dissimilarity among tasks of nodes in different clusters. This supports our approach of reducing unnecessary model training over similar tasks while addressing the non-i.i.d. cases in model reusability from different clusters. Therefore, appointing a head per cluster is deemed appropriate and effective in training tailored models when μ_{DC} is high, e.g., $\mu_{DC} > 0.7$.

4.5.3 Baselines

We conduct a comparative assessment of our DMtL with other MtL and FL models. To ensure fairness, we focus on the $\mathbf{\Omega^{-1}}$ matrix of each MtL approach under comparison, which represents the relatedness among tasks. Our DMtL is a generalization of STL; DMtL reduces to STL when the non-diagonal elements in $\mathbf{\Omega^{-1}}$ are equal to zero, i.e., $\mathbf{\Omega}_1^{-1} = \frac{1}{K}\mathbf{I}_{K\times K}$. In STL, which serves as the baseline, the models are not trained jointly. Each node trains and tunes its model locally over its own data without collaborating with other nodes.

We also compare our DMtL against the model presented in Cavallanti et al. (2010), which is an MtL perceptron for binary classification with a fixed task relationship matrix $\mathbf{\Omega}_2^{-1} = \frac{1}{K(b+1)}\mathbf{Q}_{K\times K}$. The off-diagonal and diagonal entries in $\mathbf{Q}_{K\times K}$ are set to b and $b+K$, respectively, with $b \geq 0$ as suggested in Cavallanti et al. (2010).

Moreover, we compare against the approaches in Smith et al. (2017) and Zhang and Yeung (2010), which introduce the task relationship matrix, $\mathbf{\Omega}_3^{-1}$. This matrix is updated at each training iteration with respect to model weight $\mathbf{W}$. The method in Smith et al. (2017) solves the optimization problem in (4.2) using the primal-dual method in Boyd et al. (2004), requiring a relatively large $n = \sum_{k=1}^{K} n_k$ dimensional vector, which depends on the size of *all* nodes' datasets. This size equals the total number of samples across all involved datasets needed to be exchanged among all nodes via a centralized node (e.g., Cloud data center) in each iteration. Due to the significant communication and data transfer overhead in Smith et al. (2017), the model under comparison in Zhang and Yeung (2010) adopted the methodology of the approach in Smith et al. (2017) for MtL in a centralized mode only.

Our task relationship matrix in DMtL, hereinafter denoted as $\mathbf{\Omega}_4^{-1}$ for notation compatibility with the models under comparison, is trained in a distributed SGD-based MtL fashion. $\mathbf{\Omega}_4^{-1}$ captures the PLC-driven similarity for distributed MtL with tuning parameter ϵ, whose value is discussed in Section 4.5.3. For consistency, $\mathbf{\Omega}_1^{-1}$ is associated with the STL mechanism, $\mathbf{\Omega}_2^{-1}$ refers to the Fixed Relationship (FR) method in Cavallanti et al. (2010), $\mathbf{\Omega}_3^{-1}$ is referred to as the Weighted Parameters (WP) method in Smith et al. (2017) and Zhang and Yeung (2010), and ours (PLC) is referred to as $\mathbf{\Omega}_4^{-1}$ in the experimental evaluation.

For reasons of completeness, we investigate the behavior of centralized FL (FedAvg, McMahan et al. (2017)) and decentralized FL (DFedAvg, Sun et al. (2022)) in model reusability. Both approaches train holistic models across the network. FedAvg engages a central server that aggregates local models of $O(K)$ randomly selected nodes during horizon T. At each round, each selected node i locally trains its model $\boldsymbol{\omega}_i$ using SGD for E epochs and sends it to the server, which aggregates all models $\{\boldsymbol{\omega}_i\}$ (see Appendix D). Model aggregation in FedAvg is *agnostic* to PLCs, thus, the final model is unaware of the statistical properties of nodes' data and tasks, and is unable to tackle non-i.i.d. cases. DFedAvg is a fully distributed version of FedAvg, where nodes communicate based on the adjacency matrix $\mathcal{E}$. At each round, node i sends its parameters $\boldsymbol{\omega}_i$ to its neighbors $\mathcal{N}_i$, receives, aggregates their parameters $\{\boldsymbol{\omega}_j : j \in \mathcal{N}_i\}$, and trains using SGD.

The details of hyperparameter tuning are as follows:

- *Estimation of PLCs:* Selecting an optimal PLC dimension, which represents the number of elements in the ordered index set $\mathcal{S}$, is influenced by factors such as computational costs of the models and storage capabilities of the nodes. It is essential

to consider both constraints for determining a p value. Here, $p = 6$ as suggested in Leite and Brazdil (2005). For the number of data required to calculate the PLC set $\mathcal{S}$, we ensured that $s_p \leq \frac{1}{2}n_{train}$ as in Leite and Brazdil (2005). The bootstrapping rounds are experimentally set to $L = 10$.

- *PLC-based Clustering and Model Training:* The number of clusters K in all experiments is obtained using the BIC method for GMM in the leader. All the initial local model parameters $\boldsymbol{\omega}_0$ in the CP are zero d-dimensional vectors. To fairly conduct comparative experiments with all approaches, we adopt early stopping to decide the number of training rounds T until convergence. The schedule for the learning rate $\eta_t = O(\frac{1}{t}), \eta_t \in (0, 1)$ is chosen as in Bottou (2012).

- *Hyperparameters per Model:* The decision of hyperparameters varies with the experiments and the models. For STL, we leverage the SGD classifier in *sklearn*, where the default setting α is equivalent to λ_2. Based on our experimental results, we select for DMtL the best regularization parameters $\lambda_1, \lambda_2 \in \{1e-05, 5e-05, 1e-04, 5e-04, 1e-03, 5e-03, 1e-02, 5e-02, 0.1, 0.5, 1, 5, 10\}$ and tuning parameter $\epsilon \in \{\exp(\beta) : \beta \in [-3, 6]\}$. Because all the off-diagonal entries in the FR are the same, it is not necessary to tune the parameter b in $\boldsymbol{\Omega}_2^{-1}$, which are equivalently tuned by the selection of λ_2.

4.6 Results and Analysis

4.6.1 Complexity Analysis

We first present the complexity of the proposed two-phase DMtL learning framework, and additional experiments are tested in the following sections. The clustering phase complexity is clarified as follows. The network communication cost for nodes sending their p-dimensional PLCs to the leader is $O(Mp)$. At the leader node, the Gaussian Mixture Model (GMM) used for PLC-based node grouping into K clusters requires $O(KMp^3)$ (Reynolds, 2009). The number of clusters K can be determined either by prior knowledge about tasks or by selecting the number that minimizes the Bayesian Information Criterion (BIC) (Bishop, 2007). The average pairwise similarity computation in each cluster $\mathcal{C}_k$ with M_k nodes is $O(M_k p)$, while computing PLC distances in the $\boldsymbol{\Omega}_{M\times M}$ matrix requires $O(M^2)$ time. Therefore, the overall cost for the Clustering Phase (CP) at the leader node is:

$$O\left(LM\sum_{s=1}^{p}\frac{1}{\gamma_s} + Mp\left(Kp^2 + \frac{1}{K}\right) + M^2\right). \tag{4.15}$$

As for the complexity of communication during DP, the communication cost per round t is $O(K^2 d)$, involving the exchange of $K \ll M$ model weights among head nodes in

Algorithm 7. The total communication cost is $O(TK^2d)$ for T rounds, where T depends on the convergence rate. Since $\mathbf{\Omega}^{-1}_{K\times K}$ consists of constants obtained from CP, the required T for convergence is $T = O(K\eta^*)$ with $\eta^* \in [\frac{1}{K}, 1]$ (Liu et al., 2017). The convergence rate T is linearly related to the number of tasks, and based on CP's complexity, $T = O\left(K\left(\frac{1}{\gamma} - \frac{1}{\gamma_p}\right)\right)$ for PLC-based DMtL convergence over K tasks.

4.6.2 Synthetic Dataset Experimental Evaluation

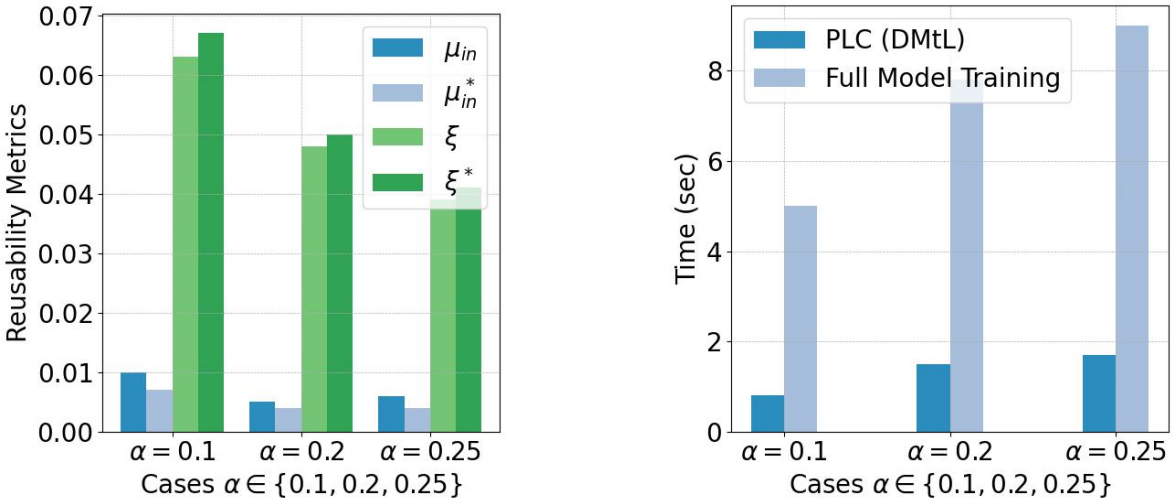
We experiment with three cases having step size $a \in \{0.1, 0.2, 0.25\}$. Each case has $\frac{1}{a} - 1$ types of combinations of assigning the $\{0, 1\}$ class labels in the data points. Note: $s_p = 500$ in SD experiments.

Figure 4.3(left) shows the model reuse metrics gained by our DMtL with $\mu_{DC} = (0.74, 0.92, 0.97)$ for $a = (0.1, 0.2, 0.25)$, respectively. As a increases, μ_{DC} increases since a larger step size means a larger discrepancy in the proportion of the binary class datasets indicating highly non-i.i.d. cases. This leads to generating more discriminated data, thus, each formed cluster driven by the models' PLCs contains relatively highly similar tasks, which tackles the non-i.i.d. challenge.

Given the three generated datasets, whose probabilities of class label 0 are $p' \in \{0.25, 0.3, 0.5\}$, we observe in Fig. 4.3(left) that the difference between the first and third datasets is larger than that of the second. GMM clustering efficiently exploits PLC information by adding similar tasks within the same clusters. This is consistent with the trend of μ_{in} (with consistently low values as a increases), indicating that PLC clustering ensures less degradation on model reuse performance within a cluster even with non-i.i.d. data.

Considering the *model reuse performance* degradation, even if we get a relatively good clustering result measured by μ_{DC} (e.g., $\mu_{DC} = 0.74$ for $a = 0.1$), our DMtL benefits from the fact that $\xi > 0$ is positive and small (see Fig.4.3(left)). Our reuse policy decreases the model reuse degradation compared with one reusing models entirely randomly. Our head selection criterion yields tailored models f^*_k reusable from any cluster member for all the cases indicated by $\mu^*_{in} < \mu_{in}$ (see Fig. 4.3(left)). This denotes that: (i) f^*_k have successfully captured all tasks/data characteristics of members within clusters, thus, decreasing the prediction loss and (ii) PLC clustering successfully separates similar from dissimilar tasks. Especially, clustering achieves very high efficiency ($\mu_{DC} > 0.9$ in cases $a \in \{0.2, 0.25\}$) having model reuse performance degradation within each cluster very low ($\mu_{in} < 0.005$).

Moreover, given that ξ and ξ^* are relatively small, positive and decrease with a, it suggests that we can always reuse models for analytics tasks from nodes belonging to different clusters even with larger data heterogeneity. For instance, given the case where $a = 0.2$ (achieving high clustering efficiency w.r.t. μ_{DC}), we obtain a relatively small degradation in the model reuse accuracy outside of any cluster ($\mu_{out} = \xi - \mu_{in} = 0.04$



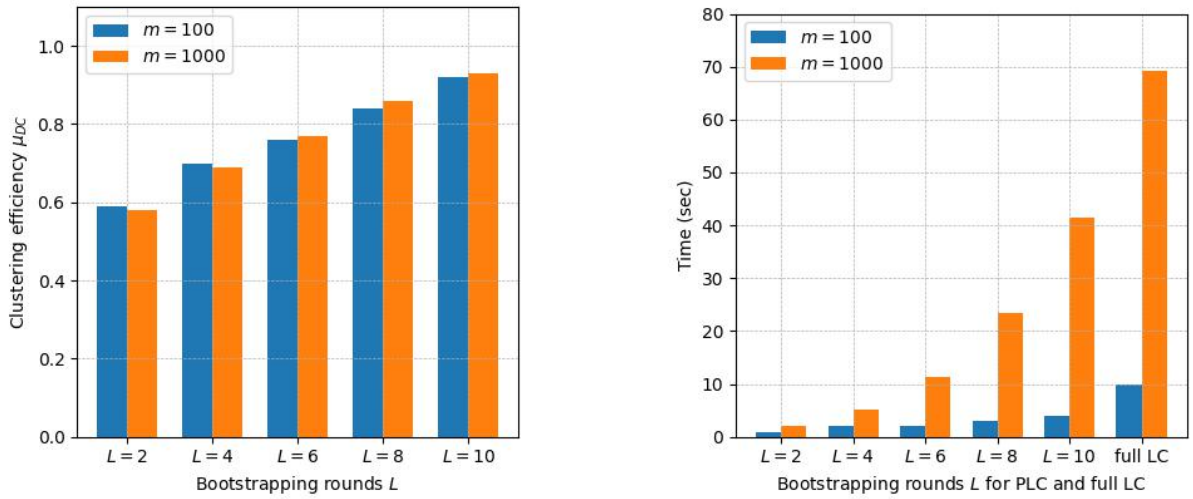
(a) (b)

Figure 4.3: SD Dataset. (left) DMtL model reusability metrics; (right) Training costs comparison between our PLC (DMtL) with $L = 4$ and full model training across all tasks for cases $a \in \{0.1, 0.2, 0.25\}$ and $M = 100$.

for $a = 0.2$). This suggests that nodes can reuse tailored models from other clusters rather than that of their head obtaining a relatively low degradation in accuracy. This signifies the effectiveness of our DMtL in reusing models over data heterogeneity within and without clusters alike.

Figure 4.3(right) shows the achieved training efficiency of our DMtL for the three cases w.r.t. a against the scenario where all nodes fully train their models (i.e., without PLC estimation and PLC clustering). The total training cost of DMtL includes the PLC estimation time, PLC clustering on the leader, and training of head nodes' tailored models. It is evident that even for classifiers/models whose complexity does not scale with the size of training data, our DMtL can still benefit from less total execution time.

Furthermore, in Fig. 4.4, we investigate the influence of the bootstrapping rounds L in PLC estimation on the training efficiency (training time) and clustering efficiency expressed by μ_{DC} for different numbers of nodes M. One can observe a trade-off between training efficiency and clustering efficiency for the case $a = 0.2$; similar results are obtained for other a values. A relatively small number of bootstrapping rounds, $L \in \{2, 4\}$, can achieve low PLC estimation time at the expense of sacrificing PLC clustering efficiency. However, for even $L \in \{6, 8\}$, we obtain an average $\mu_{DC} = 0.827$, indicating clusters of nodes with very high similar tasks. This denotes the statistical capacity and effectiveness of PLC to represent models' performances, successfully grouping nodes guaranteeing they have similar tasks with high confidence. This is the fundamental property of our DMtL to tackle data heterogeneity, aiming at learning tailored models to be reused from nodes of the same clusters and potentially from nodes outside clusters.



(a) (b)

Figure 4.4: Influence of bootstrapping rounds L on (left) μ_{DC} and (right) on training efficiency; $a = 0.2$ with nodes $M = \{100, 1000\}$.

4.6.3 Real Datasets Experimental Evaluation

We compare our DMtL against all comparison approaches in Section 4.5.3 over the benchmarks CIFAR-10 (Krizhevsky et al., 2009) and Sentiment (Blitzer et al., 2007). The corresponding results are provided in Figs. 4.5 and 4.6 and Table 4.1. Similar to Blitzer et al. (2007), in Sentiment we devise three cases of training percentages $C_p = \{0.1, 0.3, 0.5\}$ for Cases 1, 2, and 3, respectively. In CIFAR-10, we devise training percentages $C_p = \{0.01, 0.1, 0.5\}$, e.g., $C_p = 0.3$ refers to a $30\% - 70\%$ training-testing split. The results correspond to classification accuracy, reusable metrics, and communication load averaged over 100 runs for all methods STL ($\mathbf{\Omega}_1^{-1}$), FR ($\mathbf{\Omega}_2^{-1}$), WP ($\mathbf{\Omega}_3^{-1}$), PLC ($\mathbf{\Omega}_4^{-1}$), FedAvg, and DFedAvg.

One can observe from Fig. 4.5 that the average performance of PLC outperforms STL for all cases, while in Fig. 4.6, PLC achieves significantly lower inter-cluster μ_{in} and intra-cluster ξ^* reuse metrics across all methods and cases in both benchmarks. This indicates that our PLC-based DMtL achieves higher classification accuracy and effective model reusability by jointly training models across only the heads incorporating PLC information, thus avoiding redundancy due to inherent task similarity.

The tailored f_k^* are effectively reusable for other nodes compared against FR and WP due to the fact that the relationship matrices of FR and WP do not take into account any model performances of nodes, while PLC explicitly relies on these performances to selectively choose heads to learn reusable models. Especially, FR adopts a fixed task relationship matrix, whose entries do not capture any variability of the (dis)similarities of model performances across nodes, making this method unsuitable for data and task heterogeneity.

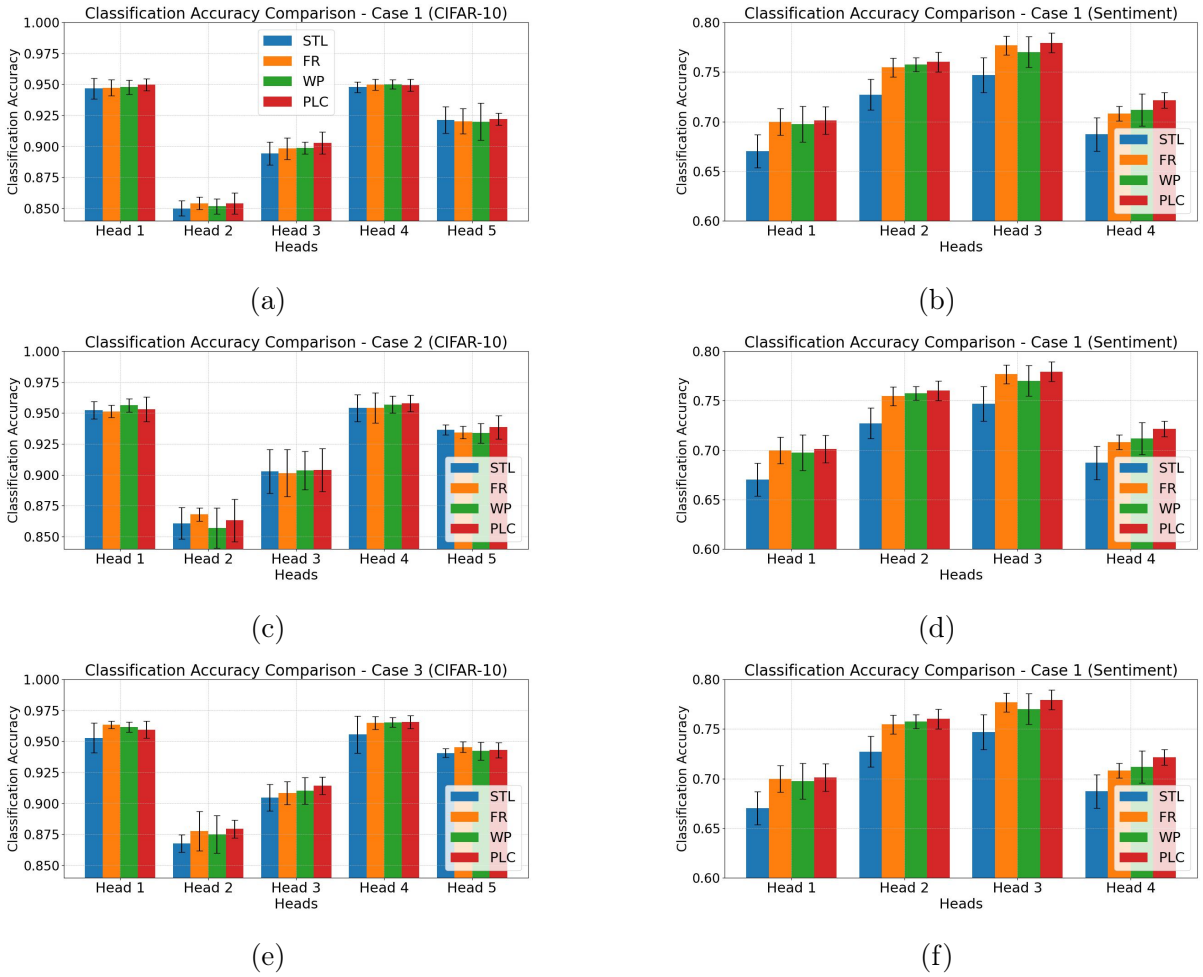


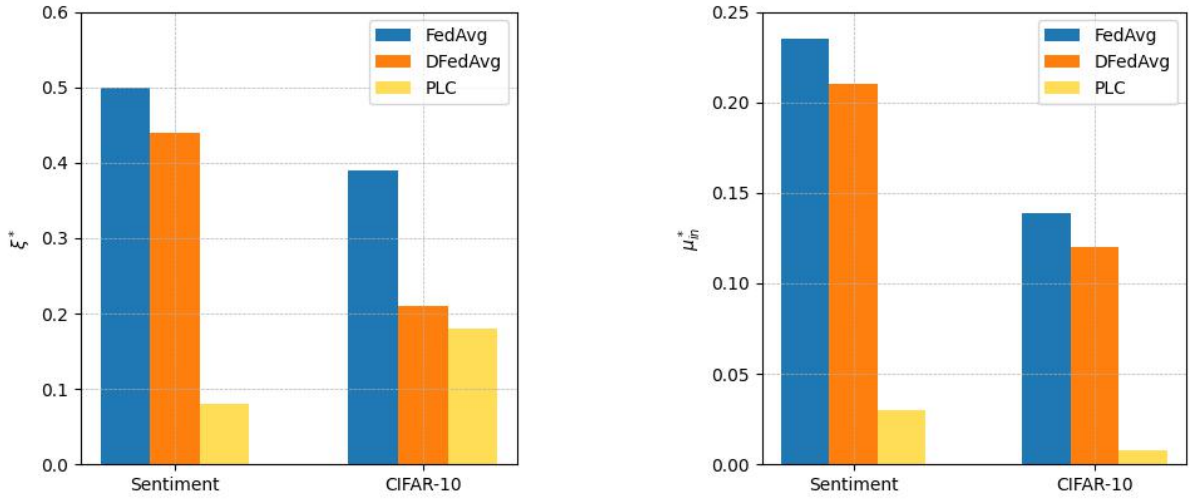
Figure 4.5: Average classification accuracy α of STL, FR, WP, and PLC on reusable tailored models; CIFAR-10 ($K = 5$); Sentiment ($K = 4$).

We also compare PLC-based DMtL with the FL-driven methods FedAvg and DFedAvg in terms of communication load (messages exchanged) to achieve convergence (adopting early stopping to all methods for fairness) and capability of reusing the optimal models assessed via ξ^* and μ^*_{in} metrics under the same setting. Table 4.1 summarizes the communication complexity and load (messages exchanged) per node until convergence for FedAvg, DFedAvg, and PLC-based DMtL, respectively, for Sentiment and CIFAR-10.

Method	Complexity	Sentiment	CIFAR-10
PLC	$O((T+1)\log K)$	33	77
DFedAvg	$O(T \log M)$	96	257
FedAvg	$O(TM)$	320	480

Table 4.1: Communication load for PLC, FedAvg, and DFedAvg.

As evidenced in Fig. 4.6, the models of FedAvg and DFedAvg are not suggested for being reused due to relatively high ξ^*. The one-model-fits-all principle of FL is unsuitable



(a) (b)

Figure 4.6: Difference in reusable model accuracy metrics (left) ξ^* and (right) μ^*_{in} for PLC, FedAvg, and DFedAvg; CIFAR-10 and Sentiment.

in model reusability due to the inherent statistical heterogeneity of tasks and data across nodes. On the contrary, PLC-based DMtL tackles such heterogeneity by building tailored models that are reused effectively, as evidenced by small ξ^* (intra-cluster) and μ^*_{in} (inter-cluster) in Fig. 4.6. PLC-based DMtL achieves lower load compared to DFedAvg because of not only involving $K < M$ heads, but these chosen heads convey the minimum sufficient knowledge to build reusable models due to PLC clustering. PLC-based DMtL is deemed appropriate for tackling the model reusability problem in an effective and efficient way and is even more favored in non-i.i.d. cases where heads hold dissimilar tasks whilst unnecessary and similar models are excluded.

4.7 Limitations and Future Research

Despite the promising outcomes of this study, several limitations need to be addressed:

Data and Task Heterogeneity: One significant limitation pertains to the varying degrees of heterogeneity across distributed nodes. Our findings suggest that while PLC clustering can enhance model reusability, its effectiveness decreases as the data becomes more homogeneous (i.e., independent and identically distributed). In scenarios where data is i.i.d., the advantages of DMtL and clustering are minimal, offering negligible improvement over traditional methods. Future research could focus on developing techniques that dynamically adjust to the level of data heterogeneity. Such methods would involve detecting heterogeneity to determine the optimal number of clusters during the clustering phase, thereby avoiding the unnecessary training of multiple models in i.i.d. scenarios.

Model Complexity and High-Dimensional Data: The models in our experiments

were classifiers applied to high-dimensional data or extracted features. The complexity of these models and the characteristics of the data can limit the tuning and overall performance of DMtL. This limitation implies that the results may not be easily generalizable to other types of models or data with different features. Future research could explore other meta-features that can be used for clustering tasks when employing deep neural networks, such as model updates or gradients. This exploration could lead to more effective clustering strategies and improved model performance.

4.8 Conclusions

In this section, we introduce a novel Distributed Multi-task Learning (DMtL) framework to address the challenges of model reusability and efficiency in edge computing environments characterized by non-i.i.d. data and constrained resources. Our approach leverages Partial Learning Curves (PLC) for clustering tasks based on performance metrics and engages selected head nodes to train tailored models, thereby enhancing generalization and reusability collaboratively.

Our extensive experimental evaluation of both synthetic and real datasets demonstrates the effectiveness of the proposed DMtL framework. Specifically, we show that our method slightly outperforms Single Task Learning (STL) and other Multi-task Learning (MtL) approaches by achieving higher classification accuracy and lower performance degradation, both within and outside clusters. The results highlight the robustness of PLC-based clustering in handling data heterogeneity and the efficiency of our two-phase framework in reducing computational and communication costs.

Our investigation into the impact of bootstrapping rounds on training efficiency and clustering effectiveness further validates the scalability of our framework. The ability to maintain high clustering efficiency with varying numbers of nodes underscores the potential of DMtL to scale in dynamic distributed environments. In conclusion, our DMtL framework presents a significant advancement in distributed learning by effectively leveraging reusable models to enhance predictive performance and resource efficiency. The proposed approach not only reduces the communication load by engaging only head nodes in the training process but also offers flexibility and scalability by seamlessly accommodating new nodes. These contributions position our framework as a valuable solution for distributed predictive modeling in edge computing environments.

The next chapter will focus on the federated learning paradigm, which supports the training of deep neural networks. Also, the target is to provide efficient solutions to reduce the computation and communication costs during training and faster inference for resource-constrained devices.

Chapter 5

Efficient Centralized Federated Learning with Pruning

5.1 Introduction

We explored distributed statistical learning techniques in previous chapters, including the concept of reuse and their applications in edge computing environments. Chapter 3 introduced methods for efficient model reuse in edge computing environments. Chapter 4 extended these concepts by presenting enhanced reusability techniques tailored for distributed learning frameworks, focusing on improving the scalability and robustness of distributed training. While these advancements improved the performance of distributed learning systems, the increasing complexity of tasks and the computational demands of neural network training necessitate further optimization strategies(Li et al., 2020c; Kairouz et al., 2021).

Federated learning (FL) has emerged as a promising paradigm to address these challenges by enabling collaborative model training across decentralized data sources while preserving data privacy (McMahan et al., 2017; Wang et al., 2020; Li et al., 2020c). However, system and data heterogeneity in FL introduces new complexities, particularly regarding computational overhead and communication costs. These issues become even more unignorable as the size and depth of neural networks increase (Konečnỳ et al., 2016).

To tackle the above challenges, this chapter proposes an efficient centralized FL framework incorporating pruning techniques to reduce computational and communication costs. Pruning, a technique for compressing neural networks, involves removing redundant parameters and structures from the model without significantly compromising its performance (Han et al., 2015). By leveraging pruning into the federated learning process, we target to balance model accuracy and efficiency (i.e., the ratio of pruned model parameters), making FL more suitable for large-scale and resource-constrained environments (Sattler et al., 2019).

In this chapter, we will:

- Provide an overview of the current state of FL and discuss compression methods for neural networks within the context of FL.
- Present our proposed framework for efficient centralized FL with extreme pruning, detailing the algorithmic strategies and convergence analysis.
- Evaluate the performance of our framework through extensive experiments, comparing it against baseline federated learning models and demonstrating its effectiveness in reducing costs while maintaining model accuracy.

Our approach aims to extend the capabilities of federated learning by leveraging pruning techniques to address the critical challenges of efficiency and scalability. By reducing the size of the neural network models during the federated learning process, we aim to facilitate a similar convergence rate and lower communication overhead, making federated learning more practical for real-world applications.

This chapter aims to offer a comprehensive solution for efficient and scalable model training in decentralized environments with pruning.

5.2 Related Work

This section starts with the necessity of federated learning and efficient distributed learning methods. Recall that the discussion about federated learning is in Chapter 2.

5.2.1 Background

As discussed in Chapter 2, traditional distributed machine learning distributes computational tasks across multiple machines or nodes, which is fundamental to achieving the same goal. However, traditional methods have become insufficient as demands evolve.

Firstly, as highlighted by Liu et al. (2022), it is challenging to bring data together. Privacy and security concerns of data, such as facial images or healthcare records, are gaining attention, resulting in impermissible data centralization. Besides, data protection regulations have been strengthened globally. For instance, the General Data Protection Regulation (GDPR) in Europe and the California Consumer Privacy Act (CCPA) in the United States set high standards for data privacy (ISACA, 2023). Additionally, China's Personal Information Protection Law (PIPL) closely follows the GDPR, underscoring the importance of personal data protection and imposing strict requirements (Law, 2023). Secondly, as the complexity of tasks increases, deep neural networks (DNNs) have proven more practical for distributed training than support vector machines (SVMs) or decision trees (Verbraeken et al., 2020).

McMahan et al. (2017) defined FL as a model that relies on a central server to aggregate model weights or gradients from various clients or nodes, offering a more advanced solution for distributed machine learning. It obeys regulations such as GDPR to ensure data security and privacy and provides a flexible framework to balance the communication and computation costs for distributed computing under DNN architectures. Since the introduction of Federated Learning (FL) by McMahan et al. (2017), the field has seen extensive research and development. Liu et al. (2022) have summarized the life cycle of FL into four distinct phases: the *composition phase*, the *training phase*, the *evaluation phase*, and the *deployment phase*. Our focus is on the **training phase**, following this life cycle to evaluate and deploy the computed models. This section provides an overview of the research work related to algorithms for federated learning. Technical details, such as formulations, will be expanded and discussed in Section 5.3.

The main challenges in federated learning can be summarized as three aspects, including *heterogeneity* (data and system), *security issues*, and *scalability*. The details have been explained in Chapter 2. To answer the research question about efficiency in Section 1, our scope is around the efficient FL.

5.2.2 Efficient Distributed Computing with Compression

Expensive and redundant sharing of model weights presents a significant challenge in distributed learning (Li et al., 2020c). Hence, recent work shows a trend in leveraging compression techniques to improve the efficiency of distributed learning. Various methods have been proposed to reduce the size of exchanged information through compression and sparsification to address this. Alistarh et al. (2018) introduced magnitude selection on model gradients to achieve sparsification in Stochastic Gradient Descent (SGD). Aji and Heafield (2017) proposed a distributed SGD method that retains only 1% of gradients by comparing their magnitudes. Similarly, Strom (2015) enhanced SGD training of DNNs by controlling the update rate per individual weight. Konečnỳ and Richtárik (2018) developed an encoding mechanism for SGD-based vectors to reduce communication overhead. Jiang and Agrawal (2018) introduced periodic quantized averaging SGD, which achieves similar predictive performance while reducing shared model gradients by 95%. Lin et al. (2018) demonstrated that 99% of gradients are redundant and proposed deep gradient compression, achieving compression rates of 270-600 without sacrificing accuracy. Building on this, the *gTop-k* gradient sparsification method by Shi et al. (2019) reduces communication costs based on the *Top-k* method. Further, Sun et al. (2019) developed a method based on Chen et al. (2018) that adaptively compresses model gradients via quantization. Reisizadeh et al. (2020) proposed FedPAQ, a framework integrating gradient sparsification and quantization to reduce communication costs. Amiri et al. (2020) combined gradient sparsification with quantized global model updates to enhance convergence. Wangni et al.

(2018) investigated error-compensated gradient sparsification, maintaining model accuracy while significantly reducing communication overhead. Abdi and Fekri (2020) introduced a compression technique based on compressed sensing for efficient gradient communication. Lastly, Han et al. (2020) developed an adaptive gradient sparsification method using an online learning approach for efficient federated learning.

Our focus is pruning since it reduces communication and computation costs without manual design. Specifically, in contrast to gradient sparsification, reducing the entire model size is crucial in distributed learning. This eliminates communication redundancy during training and enables reduced storage and inference time, enhancing the applicability of federated learning (FL) in distributed systems. Limited work has applied pruning in a distributed learning paradigm, but recent methods like Complement Sparsification (CS)(Jiang and Borcea, 2023), Federated Dynamic Sparse Training (FedDST)(Bibikar et al., 2022), and PruneFL (Jiang et al., 2022) have shown promise. CS involves collaborative pruning at the server and client levels, significantly cutting bidirectional communication costs while maintaining model performance. FedDST uses dynamic sparse training to adaptively prune models during FL, effectively addressing non-i.i.d. data distributions and reducing both communication and computation overhead. PruneFL integrates adaptive and distributed parameter pruning, dynamically adjusting model size throughout the FL process to optimize efficiency. These findings highlight the potential of pruning to boost the efficiency and scalability of FL.

5.3 Methodology

5.3.1 Preliminaries

This section brings forward the formulation of federated learning and the concepts of model pruning.

In this chapter, we adopt specific notational conventions for scalars, vectors, and matrices to ensure clarity and consistency. Scalars are denoted by lowercase letters (e.g., x), while vectors are represented by lowercase boldface letters (e.g., $\mathbf{x}$). Matrices are indicated by uppercase boldface letters (e.g., $\mathbf{X}$). Details of notations are concluded in Table 2.2 and 2.3.

For a vector $\mathbf{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d$, the ℓ_p norm ($p \geq 1$) is defined as $\|\mathbf{x}\|_p = \left(\sum_{i=1}^{d} |x_i|^p\right)^{1/p}$. The ℓ_∞ norm is defined as $\|\mathbf{x}\|_\infty = \max_{1\leq i\leq d} |x_i|$, and the ℓ_2 norm is simply $\|\mathbf{x}\|_2$. For a function $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, its gradient is represented as $\nabla f(\mathbf{x})$ and its Hessian as $\nabla^2 f(\mathbf{x})$. The minimum value of the function is denoted by $\min f$. The expectation with respect to the underlying probability space is denoted by $\mathbb{E}[\cdot]$.

Federated Learning

Consider a system consisting of a set of M clients and one central server. Assume that the task is for supervised learning. Let matrix $\mathcal{D}_i = \{(\mathbf{x}, y)\}^{n_i \times (d+1)}$ denote the data owned by client i with n_i data samples, where $i \in |M|$ with $|M| = \{1, 2, \ldots, M\}$, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{n_i \times d}$, and $y \in \mathcal{Y} \subset \mathbb{R}^{n_i}$.

To avoid the stragglers problem, FedAvg (McMahan et al., 2017) select out a subset of $\mathcal{C} \subset \mathcal{M}$ clients for collaborative training with a central server, where $|\mathcal{C}| = C$. Then the local loss is computed:

$$f_i(\boldsymbol{\omega}) = \frac{1}{n_i} \sum_{(\mathbf{x},y)\in\mathcal{D}_i} \mathcal{L}(\boldsymbol{\omega}^\top \mathbf{x}, y) \tag{5.1}$$

where $\boldsymbol{\omega}$ is the model parameter and $\mathcal{L}$ is a loss function that measures the quality of the prediction. Then the global optimization function for all the selected nodes $\mathcal{C}$ is:

$$f(\boldsymbol{\omega}) = \sum_{i\in\mathcal{C}} \rho_i f_i(\boldsymbol{\omega}), \text{ where } \rho_i = \frac{n_i}{\sum_{i\in\mathcal{C}} n_j}. \tag{5.2}$$

The optimization process is continuous over T global rounds with E_l local epochs. Let $t \in \{0, 1, \ldots, T-1\}$ denote a discrete-time instance during the training process. Then, $\tau = \left\lfloor \frac{t}{E_l} \right\rfloor L$ is the start time of the current global epoch. At τ, the clients receive updated aggregated weights $\bar{\boldsymbol{\omega}}^\tau$ from the server node responsible for aggregating the clients' model parameters. The local training at client m during local epoch $l = 1, \ldots, E_l$ proceeds as follows:

$$\boldsymbol{\omega}_i^{(\tau+l)+1} = \boldsymbol{\omega}_i^{\tau+l} - \eta_{\tau+l} \nabla f_i(\boldsymbol{\omega}_i^{\tau+l}), \tag{5.3}$$

where $\eta_{\tau+l} \in (0, 1)$ is the learning rate. The aggregation policy on the side of the central server can be expressed as:

$$\bar{\boldsymbol{\omega}}^\tau = \sum_{i\in\mathcal{C}} \rho_i \boldsymbol{\omega}_i^\tau. \tag{5.4}$$

An instance of FL framework is shown in Figure 5.1.

Model Pruning

In centralized learning systems, where all data is stored centrally, pruning aims to sparsify the weight matrices of deep neural network (DNN) models by setting a significant proportion of weights to zero while maintaining model performance (Zhu and Gupta, 2017). Pruning techniques can be classified as either *structured* or *unstructured* depending on whether entire sub-structures, such as channels, filters, and layers, are pruned. Another important concept is *sparsity*, denoted as $s \in [0, 1]$, which represents the proportion of non-zero weights in the model. For instance, a model with $s = 1$ is fully sparse with all weights equal to zero, whereas $s = 0$ represents the original model with no pruning. Figure

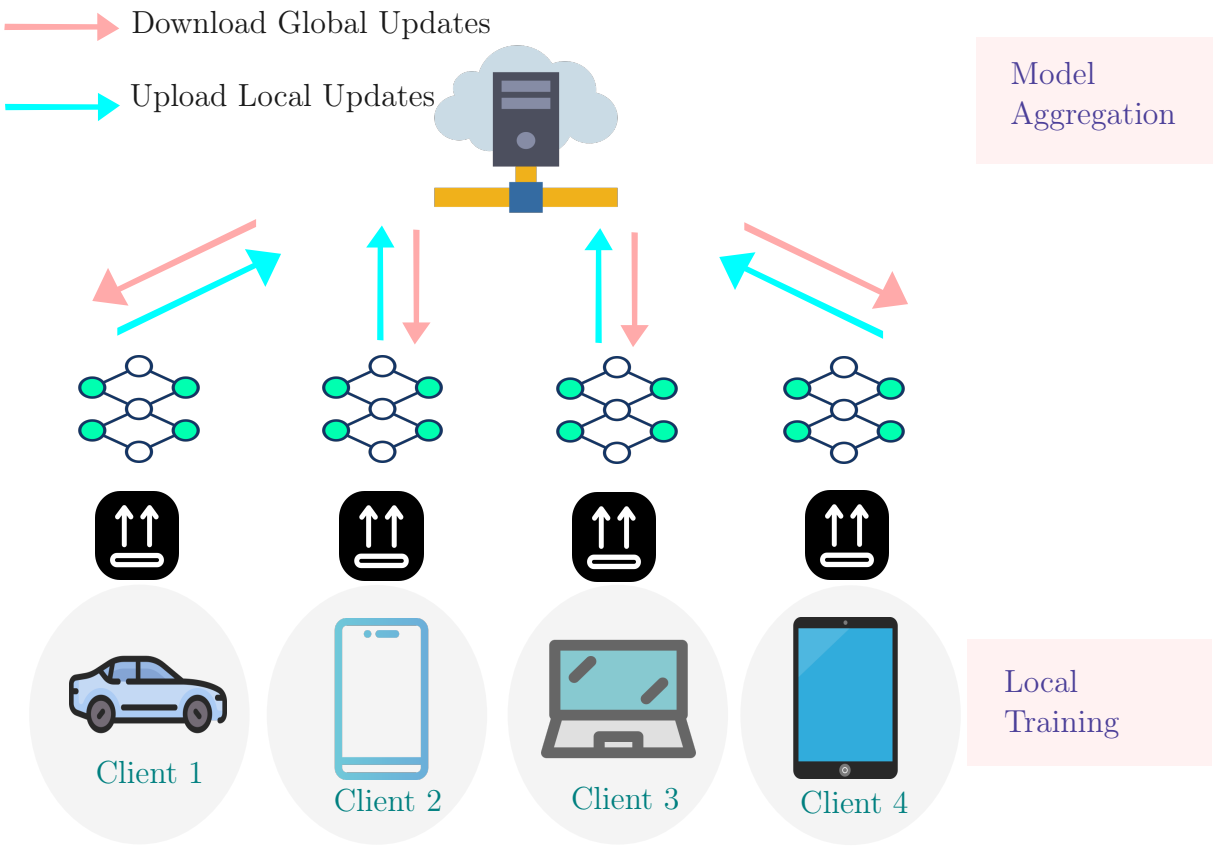


Figure 5.1: Federated Learning Framework Diagram

5.2 depicts one example of pruning for neural networks.

Pruning typically involves using mask functions that act as indicator functions, deciding whether a weight should be zero based on a criterion, often the absolute value of the weight. Weights below a predefined threshold are pruned, effectively reducing the model's complexity and improving computational efficiency.

In federated learning (FL), pruning is essential for reducing communication costs during training. High sparsity, $s \geq 0.8$ (Hoefler et al., 2021), are desirable to minimize communication overhead while ensuring that the global model remains effective with only a marginal loss in prediction accuracy. Distributed and adaptive pruning methods are employed to achieve these high compression rates, making FL more feasible and efficient (Jiang et al., 2022).

According to pruning time, pruning techniques can be categorized into three main types:

- **Pruning Before Training**: Techniques such as SNIP (Lee et al., 2018) determine important weights before the actual training begins.
- **Pruning During Training**: Methods like PruneTrain (Lym et al., 2019), DPF (Lin et al., 2020), and FedDST (Bibikar et al., 2022) prune weights iteratively as the

training progresses. This category also includes PruneFL (Jiang et al., 2022), which integrates pruning into the FL process to enhance efficiency.

- **Pruning After Training**: This approach is less useful in distributed settings as it requires centralized retraining to fine-tune the pruned model.

Among the most common techniques are:

- **Regularization-based Pruning (RP)**: Utilizes sparsity-inducing properties of norms like L_1 (Manhattan distance) and L_2 (Euclidean distance) to limit the importance of parameters, effectively pushing unimportant weights towards zero during training.

- **Importance-based Pruning (IP)**: Prunes parameters based on their significance, determined by predefined criteria such as weight magnitude or contribution to overall model performance (Wang et al., 2021).

While RP techniques are generally superior due to their ability to induce sparsity adaptively, they face challenges in controlling the exact sparsity level and dynamically tuning the regularization parameter λ. Improper tuning can lead to excessive pruning or insufficient sparsity, affecting model performance (He et al., 2017; Zhang et al., 2018).

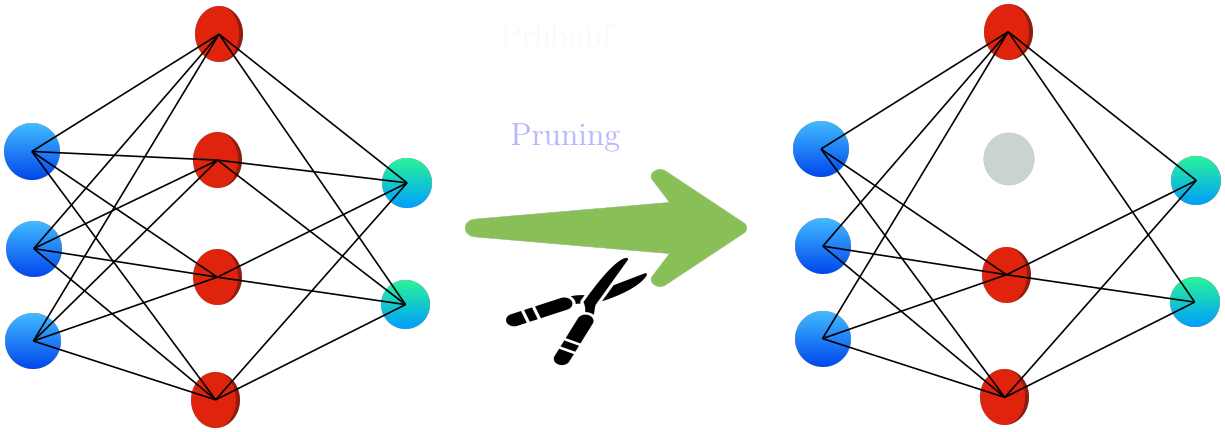


Figure 5.2: **Example of Pruning:** The diagram illustrates the pruning process in a neural network model. It shows the connections between different nodes and layers, highlighting the paths that remain after pruning. The gray circles represent the pruned nodes, indicating the elements that have been removed to optimize the network. The remaining paths, represented by lines connecting various elements, show the network structure after pruning.

5.3.2 FedDIP Algorithm

The proposed FedDIP framework integrates extreme dynamic pruning with error feedback and incremental regularization in distributed learning environments. Figure 5.3 provides a schematic representation of FedDIP, which will be elaborated upon in this section.

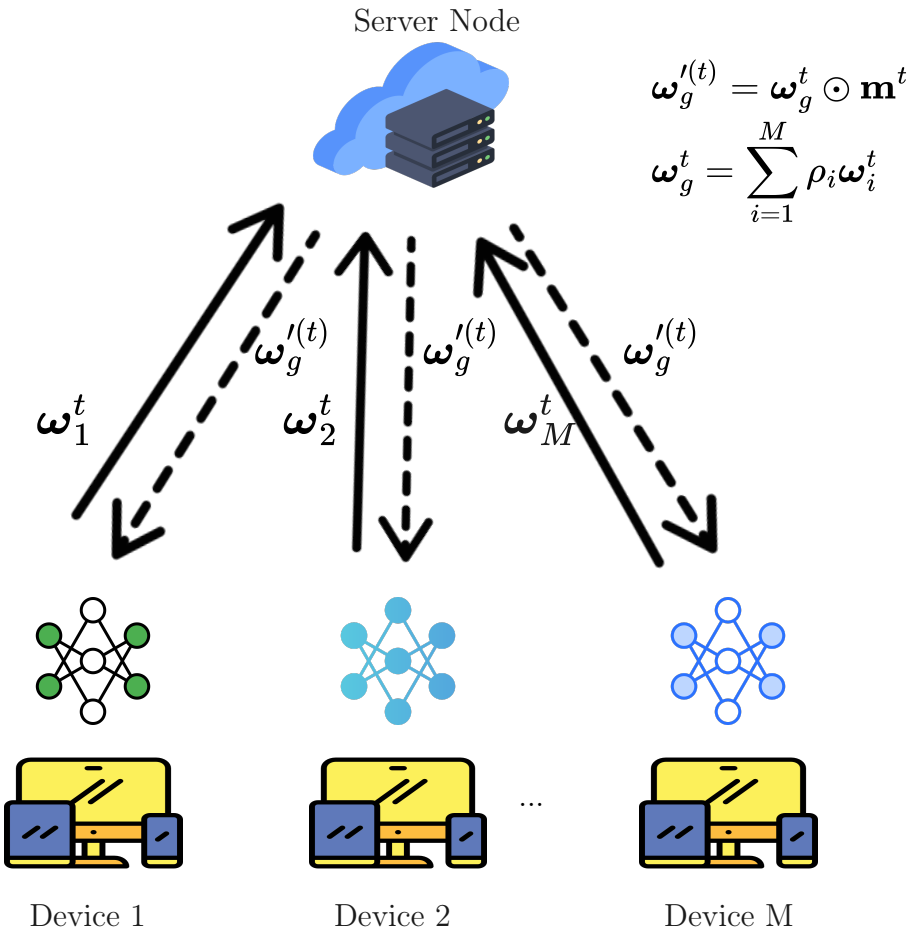


Figure 5.3: **FedDIP Framework:**
(1) *Downlink Phase:* The server node broadcasts the pruned global model ($\omega_g'^{(t)}$) to all participating clients (Devices 1 to M).
(2) *Uplink Phase:* Each selected client sends its local dense model (ω_i^t) back to the server for aggregation. The global model (ω_g^t) is updated based on the aggregation of these local models, using the formula $\omega_g^t = \sum_{i=1}^{M} \rho_i \omega_i^t$.
(3) *Sparse Training Phase:* The global mask ($\mathbf{m}^t$), derived from the global model, guides the distributed pruning and fine-tuning (DPF) process across the clients. This ensures efficient and effective sparse training.

Dynamic Pruning with Error Feedback

The dynamic pruning method (DPF) introduced by Lin et al. (2020) demonstrates improved performance under high sparsity. Given the SGD update scheme, the model gradient in DPF is computed on the pruned model as:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}^t - \eta_t \nabla f(\boldsymbol{\omega}^{t(\prime)}) = \boldsymbol{\omega}^t - \eta_t \nabla f(\boldsymbol{\omega}^t \odot \mathbf{m}_t), \tag{5.5}$$

considering the error feedback:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}^t - \eta_t \nabla f(\boldsymbol{\omega}^t + \mathbf{e}_t), \tag{5.6}$$

where $\mathbf{e}_t = \boldsymbol{\omega}^{t(\prime)} - \boldsymbol{\omega}^t$. In (5.5), $\odot$ represents the Hadamard (element-wise) product, $\boldsymbol{\omega}^t$ are the model parameters, $\boldsymbol{\omega}^{t(\prime)}$ are the pruned model parameters, and $\mathbf{m}$ is the pruning mask function used in Jiang et al. (2022), Lym et al. (2019), and Lin et al. (2020). The mask eliminates weights based on their magnitude, producing the pruned $\boldsymbol{\omega}^{t(\prime)}$. This gradient application helps recover from errors due to premature masking of important weights, ensuring the update rule in (5.5) best suits the pruned model.

In contrast, other pruning methods in FL, such as Jiang et al. (2022), adopt the rule:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}^{t(\prime)} - \eta_t \nabla f(\boldsymbol{\omega}^{t(\prime)}). \tag{5.7}$$

The update rule in (5.5) retains more information by only computing gradients of the pruned model, which is expected to yield superior performance under high sparsity.

Incremental Regularization

Wang et al. (2021) proves that growing regularization, the proposed GReg algorithm, benefits pruning. However, our approach significantly differs from Wang et al. (2021). While GReg applies structured pruning only on convolutional layers and uses growing regularization during the pruning of pre-trained models, we adopt incremental regularization during model training and apply unstructured pruning to both convolutional and fully connected layers.

Learning Framework

FedDIP combines dynamic pruning and incremental regularization to maintain predictive model performance under extreme sparsity. We denote our algorithm directly applying dynamic pruning as 'FedDP' and the variant adding incremental regularization as 'FedDIP.' The FedDIP process is summarized in Algorithm 8, where only pruned models are exchanged from the server to nodes, while clients train sparse models, saving computational costs.

Each node $m \in \mathcal{M}$ first trains a local sparse DNN model with weights of relatively small magnitudes. Then, node n optimizes the *local incrementally regularized loss function* at round t:

$$f_i(\boldsymbol{\omega}^t) = \frac{1}{n_i} \sum_{(\mathbf{x},y)\in\mathcal{D}_i} \mathcal{L}(\boldsymbol{\omega}^{(t)\top}\mathbf{x}, y) + \lambda_t \sum_{b=1}^{B} \|\boldsymbol{\omega}^{t(b)}\|_2, \tag{5.8}$$

where λ_t controls the degree of model shrinkage and B is the number of DNN layers.

Algorithm 8 The FedDIP Algorithm

Input: M nodes; T global rounds; E_l local rounds; initial and target sparsity s_0 and s_p; maximum regularization $\lambda_{\max}$; quantization step Q; reconfiguration horizon R
Output: Global pruned DNN model weights $\boldsymbol{\omega}'_g$

// **Server initialization**
if $s_0 > 0$ **then**
  Server initializes global mask $\mathbf{m}_0$ (ERK distribution)
end if
// **Node update and pruning**
for global round $\tau = 1$ to T **do**
  Server randomly selects C nodes $\mathcal{C} \subset \mathcal{M}$
  for all selected node $i \in \mathcal{C}$ **in parallel do**
    Receive pruned weights $\boldsymbol{\omega}_g^{\prime(\tau-1)}$ from server node
    Obtain mask $\mathbf{m}_{\tau-1}$ from $\boldsymbol{\omega}_g^{\prime(\tau-1)}$
    Train $\boldsymbol{\omega}_i^{\tau}$ over E_l rounds on data $\mathcal{D}_i$ using (5.10)
    if incremental regularization is chosen **then**
      Optimize (5.8) with incremental λ_τ in (5.9)
    else
      Optimize (5.1)
    end if
  end for
  // **Server update, aggregation, and reconfiguration**
  Server receives models and aggregates $\boldsymbol{\omega}_g^{\tau}$ in (5.13)
  if $\tau \mod R == 0$ **then**
    Reconfigure global mask $\mathbf{m}_\tau$ based on pruning $\boldsymbol{\omega}_g^{\tau}$
  end if
  Server prunes global model with $\mathbf{m}_\tau$ and obtains $\boldsymbol{\omega}_g^{\prime(\tau)}$
  Server node returns $\boldsymbol{\omega}_g^{\prime(\tau)}$ to all nodes
end for

The incremental regularization over λ_t follows the schedule:

$$\lambda_t = \begin{cases} 0 & \text{if } 0 \leq t < \frac{T}{Q} \\ \frac{\lambda_{\max} \cdot (j-1)}{Q} & \text{if } \frac{(j-1)T}{Q} \leq t < \frac{iT}{Q} \\ \frac{\lambda_{\max}(Q-1)}{Q} & \text{if } \frac{(Q-1)T}{Q} \leq t \leq T \end{cases} \tag{5.9}$$

with quantization step size $Q > 0$. The regularization parameter space is divided from $\frac{\lambda_{\max}}{Q}$ to $\lambda_{\max}$, gradually increasing regularization every $\frac{T}{Q}$ rounds. Each node n dynamically prunes its local model weights $\boldsymbol{\omega}_i^{\tau+L}$ to optimize (5.8) as:

$$\boldsymbol{\omega}_i^{\tau,l+1} = \boldsymbol{\omega}_i^{\tau,l} - \eta_\tau \nabla f_i(\boldsymbol{\omega}_i'^{(\tau,l)}), \tag{5.10}$$

where $\boldsymbol{\omega}_i'^{(\tau+l)}$ is pruned based on a global mask function $\mathbf{m}_\tau$ generated by the server.

Our gradual pruning policy modifies the sparsity update from Li et al. (2020d) by incrementally updating the sparsity as:

$$s_t = s_p + (s_0 - s_p)\left(1 - \frac{t}{T}\right)^3, \tag{5.11}$$

where s_t represents the sparsity at round t, s_0 is the initial sparsity, and s_p is the target sparsity. In our approach, s_0 is strictly non-zero, incrementing sparsity from moderate to extreme levels. If $s_0 > 0$, the initial mask's layer-wise sparsity follows the ERK distribution as described in Evci et al. (2020). Specifically, the sparsity s_0^b for layer b is given by

$$s_0^b \simeq \frac{n^{b-1} + n^b + w^b + h^b}{n^{b-1} \times n^b \times w^b \times h^b}, \tag{5.12}$$

,where n^b denotes the number of neurons, w stands for the width, and h represents the height. The procedure of our proposed pruning policy is shown in Figure 5.4.

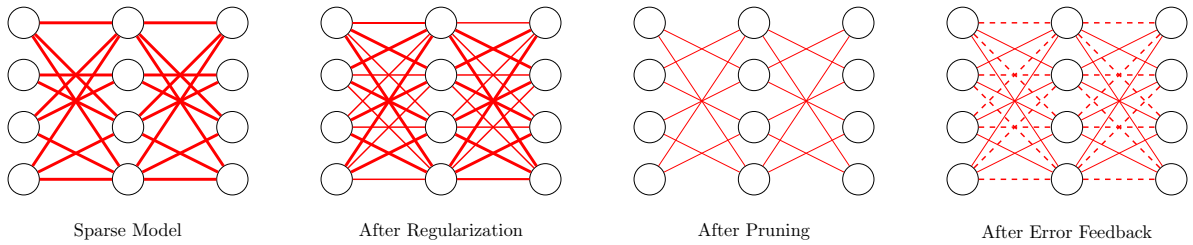


Figure 5.4: FedDIP achieves extreme sparsity through a process that begins with (1) an initial sparse model, followed by periodic pruning that incorporates (2) incremental regularization and (3) error feedback. The width of the red lines indicates the magnitude of the connections, while the dotted lines represent connections recovered through error feedback.

At the end of a local epoch E_l, the central server collects C model weights $\boldsymbol{\omega}_i^{\tau+l}$ from selected nodes $i \in \mathcal{C}$ and calculates the global weights average as:

$$\bar{\boldsymbol{\omega}}_g^{\tau+l} = \sum_{i \in \mathcal{C}} \rho_i \boldsymbol{\omega}_i^{\tau+l}. \tag{5.13}$$

The mask function $\mathbf{m}_\tau$ is generated based on pruning $\bar{\boldsymbol{\omega}}_g^{\tau+l}$ with current sparsity s_τ. FedDIP achieves data-free initialization and generalizes the DPF Lin et al. (2020) in the dynamic pruning process. When $s_0 = 0$ and no incremental regularization ($\lambda_t = 0$, $\forall t$), FedDIP reduces to distributed version of DPF.

5.4 Theoretical Analysis

This section provides theoretical support for the proposed FedDIP algorithm, including a convergence analysis. We present the assumptions and Theorem 2.

At each global round $t \in \{1, \ldots, T\}$, C out of M clients participate, each selected with probability ρ_i as in Haddadpour and Mahdavi (2019); Li et al. (2019b), ensuring $\sum_{i=1}^{M} \rho_i = 1$.

Let $\boldsymbol{\omega}_i^t$ and $\boldsymbol{\omega}_i^{\prime(t)}$ be the weights and pruned weights at round t on client i, respectively:

$$\boldsymbol{\omega}_i^{\prime(t)} = \boldsymbol{\omega}_i^t \odot \mathbf{m}^t. \tag{5.14}$$

Let $\mathbf{v}_i^t$ and $\tilde{\mathbf{v}}_i^t$ be the expected and estimated gradients at t, respectively, on client i. Based on $\boldsymbol{\omega}_i^{\prime(t)}$, we have:

$$\mathbf{v}_i^{\prime(t)} = \nabla f(\boldsymbol{\omega}_i^{\prime(t)}), \tag{5.15}$$

and $\tilde{\mathbf{v}}_i^{\prime(t)}$ is the estimated gradient. The global aggregated model for FedAvg is:

$$\bar{\boldsymbol{\omega}}^t = \frac{1}{C} \sum_{i \in \mathcal{C}} \boldsymbol{\omega}_i^t, \tag{5.16}$$

which is pruned before being sent by the server as:

$$\bar{\boldsymbol{\omega}}^{\prime(t)} = \frac{1}{C} \sum_{i \in \mathcal{C}} \boldsymbol{\omega}_i^t \odot \mathbf{m}^t. \tag{5.17}$$

The global estimated aggregated gradient and expected global gradient are:

$$\tilde{\mathbf{v}}^t = \frac{1}{C} \sum_{i \in \mathcal{C}} \tilde{\mathbf{v}}_i^t, \quad \bar{\mathbf{v}}^t = \frac{1}{C} \sum_{i \in \mathcal{C}} \mathbf{v}_i^t. \tag{5.18}$$

For DPF, we have:

$$\tilde{\mathbf{v}}'^{(t)} = \frac{1}{C}\sum_{i\in\mathcal{C}} \tilde{\mathbf{v}}_i'^{(t)}, \quad \bar{\mathbf{v}}'^{(t)} = \frac{1}{C}\sum_{i\in\mathcal{C}} \mathbf{v}_i'^{(t)}. \tag{5.19}$$

In FedAvg, $\bar{\boldsymbol{\omega}}^t$ is updated as:

$$\bar{\boldsymbol{\omega}}^{t+1} = \bar{\boldsymbol{\omega}}^t - \eta_t \tilde{\mathbf{v}}^t, \tag{5.20}$$

while the update rule based on DPF at client i is:

$$\boldsymbol{\omega}_i^{t+1} = \boldsymbol{\omega}_i^t - \eta_t \tilde{\mathbf{v}}_i'^{(t)}, \tag{5.21}$$

where $\boldsymbol{\omega}_i^t = \bar{\boldsymbol{\omega}}'^{(t)}$. Similarly, $\bar{\boldsymbol{\omega}}^{t+1}$ is updated as:

$$\bar{\boldsymbol{\omega}}^{t+1} = \bar{\boldsymbol{\omega}}'^{(t)} - \eta_t \tilde{\mathbf{v}}'^{(t)}. \tag{5.22}$$

Definition 1. *The quality of pruning is defined by the parameter $\delta_t \in [0,1]$ as:*

$$\delta_t := \frac{\|\boldsymbol{\omega}^t - \boldsymbol{\omega}'^{(t)}\|_F^2}{\|\boldsymbol{\omega}^t\|_F^2}, \tag{5.23}$$

where $\|\cdot\|_F^2$ is the square of the Frobenius matrix norm. δ_t indicates the degree of information loss by pruning in terms of magnitude, with a smaller δ_t representing less information loss.

Definition 2. *Following Wan et al. (2021a), a measurement γ of non-i.i.d. (non-independent and identically distributed) data is defined as:*

$$\gamma = \frac{\sum_{i=1}^M p_i \|\nabla f_i(\boldsymbol{\omega})\|^2}{\|\sum_{i=1}^M p_i \nabla f_i(\boldsymbol{\omega})\|^2}, \tag{5.24}$$

with $\gamma \geq 1$; $\gamma = 1$ holds in the i.i.d. case.

We list our assumptions for proving the convergence of FedDIP during the learning phase.

Assumption 1. ***L-Smoothness**. For all $\boldsymbol{\omega}^{t_1}, \boldsymbol{\omega}^{t_2} \in \mathbb{R}^d$ and $L \in \mathbb{R}$:*

$$f(\boldsymbol{\omega}^{t_1}) \leq f(\boldsymbol{\omega}^{t_2}) + (\boldsymbol{\omega}^{t_1} - \boldsymbol{\omega}^{t_2})^\top \nabla f(\boldsymbol{\omega}^{t_2}) + \frac{L}{2}\|\boldsymbol{\omega}^{t_1} - \boldsymbol{\omega}^{t_2}\|^2. \tag{5.25}$$

Assumption 2. ***μ-Lipschitzness**. For all $\boldsymbol{\omega}^{t_1}, \boldsymbol{\omega}^{t_2} \in \mathbb{R}^d$ and $\mu \in \mathbb{R}$:*

$$\|f(\boldsymbol{\omega}^{t_1}) - f(\boldsymbol{\omega}^{t_2})\| \leq \mu \|\boldsymbol{\omega}^{t_1} - \boldsymbol{\omega}^{t_2}\|. \tag{5.26}$$

Assumption 3. ***Bounded variance for gradients***. *Following Li et al. (2019b), the local model gradients on each client i are self-bounded in variance:*

$$\mathbb{E}[\|\tilde{\mathbf{v}}_i^t - \mathbf{v}_i^t\|^2] \leq \sigma_i^2. \tag{5.27}$$

Assumption 4. ***Bounded weighted aggregation of gradients***. *Following Wan et al. (2021a), the aggregation of local gradients at time t is bounded as:*

$$\|\sum_{i=1}^{M} \rho_i \mathbf{v}_i^t\|^2 \leq G^2, \tag{5.28}$$

where $\sum_{i=1}^{M} \rho_i = 1$ and $\sum_{i=1}^{M} \rho_i \mathbf{v}_i^t$ stands for the weighted aggregation of local gradients; $G \in \mathbb{R}$.

We provide the convergence of FedDIP with the following theorem.

Theorem 2 (FedDIP Convergence)**.** *Under Assumptions 1, 2, 3, and Lemmas 3, 4, 5, with $\eta_t \leq \frac{1}{tL}$, $L > 0$, the convergence rate of FedDIP is bounded by:*

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 \;\leq\; 2L\mathbb{E}(f(\boldsymbol{\omega}_1) - f^*) + 2L\sum_{t=1}^{T}[\mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|] + \frac{\pi^2}{3L^2}\chi], \tag{5.29}$$

where $f(\boldsymbol{\omega}_1)$ and f^* denote the initial and final convergent loss, respectively, with $\chi = \frac{(\gamma-1)L^2+L}{2C}\sum_{i=1}^{M}\rho_i\sigma_i^2 + \frac{(\gamma-1)\gamma E_l^2 L^2 G^2}{2}$, and γ is defined in Definition 2.

Proof. See Appendix A.3.1. □

Corollary 1. *Under the conditions of Theorem 2, with a constant learning rate η, the convergence rate of* $\min \|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2$ *is $\mathcal{O}(\frac{1}{T})$, assuming $\delta_t = \mathrm{o}(\frac{1}{T})$.*

Proof. See Appendix A.3.2. □

5.5 Experiment Setup

5.5.1 Datasets and Models

We conducted experiments using the *Fashion-MNIST* (Xiao et al., 2017), *CIFAR-10*, and *CIFAR-100* (Krizhevsky et al., 2009) datasets. *Fashion-MNIST* comprises 60,000 training images and 10,000 test images, all 28x28 grayscale images classified into 10 categories. Both *CIFAR* datasets contain 50,000 training images and 10,000 test images of 32x32 color images; *CIFAR-10* has 10 classes (6000 images per class) while *CIFAR-100* has 100 classes (600 images per class). We consider the i.i.d. (independent and identically distributed) case for comparing algorithms and extending FedDIP to non-i.i.d. scenarios.

To evaluate the performance of FedDIP, we utilized several well-known CNN architectures as the backbone (dense or unpruned) models: *LeNet-5* (LeCun et al., 2015), *AlexNet* (Krizhevsky, 2014), and *ResNet-18* (He et al., 2016).

5.5.2 Baselines

We compared the performance of FedDIP with the following baselines:

- **FedAvg** (McMahan et al., 2017): It is the standard federated learning algorithm, and details can be referred to Section 2.3.2. Since pruning is not considered, the models are dense during the communication and training process.
- **PruneFL** (Jiang et al., 2022): A federated learning algorithm that incorporates pruning during training. PruneFL is the first work to integrate adaptive and parameter pruning with federated learning, initially pruning the model at the central server or a trusted client. This requires several rounds to train the central model from dense to sparse. Further pruning is based on the importance score calculated by gradient magnitude using Taylor expansion and time sensitivity.
- **PruneTrain** (Lym et al., 2019): A pruning algorithm that prunes during training and incorporates regularization. PruneTrain is a centralized pruning technique that accelerates training through a cost-efficient mechanism that gradually reduces training costs. The group-regularization lasso aids the pruning method in reconfiguring high-accuracy models.
- **FedDST** (Bibikar et al., 2022): A dynamic sparse training method for federated learning. Unlike PruneFL, FedDST avoids the initial training to obtain the masks; instead, the mask is initialized using the ERK distribution as stated in Equation 5.12. During client training, each client performs layerwise pruning periodically to update the mask through weight drop and regrowth, similar to RigL (Evci et al., 2020).
- **DPF** (Lin et al., 2020): A dynamic pruning method with error feedback, used as a baseline in our comparisons. DPF creates a sparse model without additional overhead by enabling (i) dynamic allocation of the sparsity pattern and (ii) incorporating a feedback signal to reconnect prematurely pruned weights, resulting in an efficient sparse model in a single training pass.
- **SNIP** (Lee et al., 2018): A single-shot network pruning method that identifies and removes less important weights before training. SNIP computes the importance of each weight based on the sensitivity of the loss function to changes in the weight.

Weights with the lowest importance scores are pruned in a single step, leading to a more efficient training process with reduced model size and computational cost.

For the non-i.i.d. case, we adopted the pathological data partitioning method from (McMahan et al., 2017), assigning only two classes per node. Additionally, we combined FedDIP with FedProx (Li et al., 2020d)—a generalization and re-parametrization of FedAvg to handle data heterogeneity—termed as FedDIP+Prox. We compared this combination with the baseline FedAvg and FedProx.

5.5.3 Configurations

Table 5.1 outlines our experimental configurations. For PruneFL, FedDST, and PruneTrain, we empirically determined the optimal reconfiguration intervals R to be 20, 20, and 1 respectively, to ensure optimal model performance. The same applies to the step size Q for all models, with an annealing factor of 0.5 for FedDST. As SNIP prunes the model prior to training, the global mask achieves the target sparsity s_p via one-shot pruning. We employed grid-search to set the penalty factor for PruneTrain, ranging from 10^{-1} to 10^{-5}, depending on the experiment. Other hyperparameters were adjusted to match our settings where necessary. For the non-i.i.d. case, the penalty for the proximal term in FedProx was determined via grid search within the same range. FedDIP+Prox adopts the optimal combination of penalty values for both FedDIP and FedProx.

Our experiments were designed to comprehensively evaluate FedDIP in various FL environments, focusing on its adaptability to extreme sparsity while maintaining high model performance.

Datasets	**Fashion-MNIST**	**CIFAR-10**	**CIFAR-100**
Model	LeNet-5	AlexNet	ResNet-18
Pruning Layers (Z)	5	8	18
Learning Rate (η_0)	0.01	0.1	0.1
Clients per Round (K)	5/50	5/50	5/50
Batch Size	64	128	128
Initial Sparsity (s_0)	0.5	0.5	0.05
Global Rounds (T)	1000	1000	1000
Reconfiguration Interval (R)	5	5	5
Step Size (Q)	10	10	10
Local Rounds (E_l)	5	5	5
Max Penalty ($\lambda_{\max}$)	10^{-3}	10^{-3}	$5 \cdot 10^{-3}$

Table 5.1: FedDIP Configuration Table

5.6 Experimental Analysis

5.6.1 Performance Evaluation

To evaluate the performance of FedDIP and other baseline methods under extreme sparsity, we set the target sparsity $s_p = 0.9$ for *Fashion-MNIST* and *CIFAR-10*, and $s_p = 0.8$ for *CIFAR-100*. Due to training divergence with SNIP at $s_p = 0.9$ for *AlexNet*, we adjust s_p to 0.8 for SNIP in this case.

Accuracy

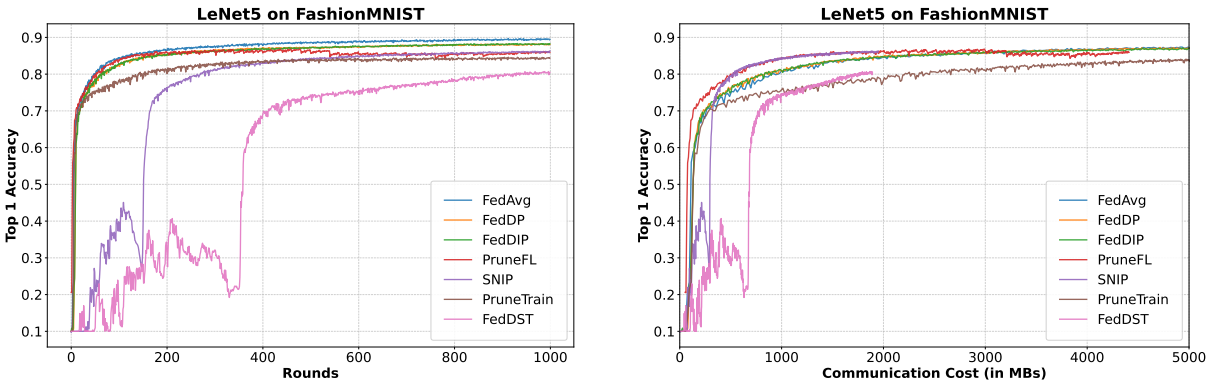
As clarified in the baseline discussion, FedAvg serves as the baseline with a dense model. Beyond FedAvg, Figures 5.5a, 5.6a, and 5.7a show that FedDIP consistently achieves the highest top-1 accuracy compared to other baseline methods under extreme sparsity. While the performance is comparable during the initial training phases when sparsity levels are low, FedDIP's advantage becomes more explicit as sparsity increases. Table 5.2 shows that FedDIP only slightly reduces the accuracy of *LeNet-5* and *ResNet-18* by 1.24% and 1.25%, respectively, at $s_p = 0.9$ and $s_p = 0.8$. For *AlexNet*, FedDIP maintains and even improves performance by 0.7% compared to FedAvg at $s_p = 0.9$, highlighting its robustness and efficiency.

Cumulative Communication and Training Cost

Efficient communication is a key factor in the success of distributed learning systems, particularly in bandwidth-constrained environments like edge computing and federated learning. To evaluate our approach under realistic constraints, we focus on balancing communication cost and model accuracy. With a fixed budget, we analyze the relationship between communication cost and accuracy. Figures 5.5b, 5.6b, 5.7b, and Table 5.3 provide an overview. The results show that FedDIP effectively prunes the model across all experiments and outperforms other methods, maintaining communication efficiency comparable to other pruning methods.

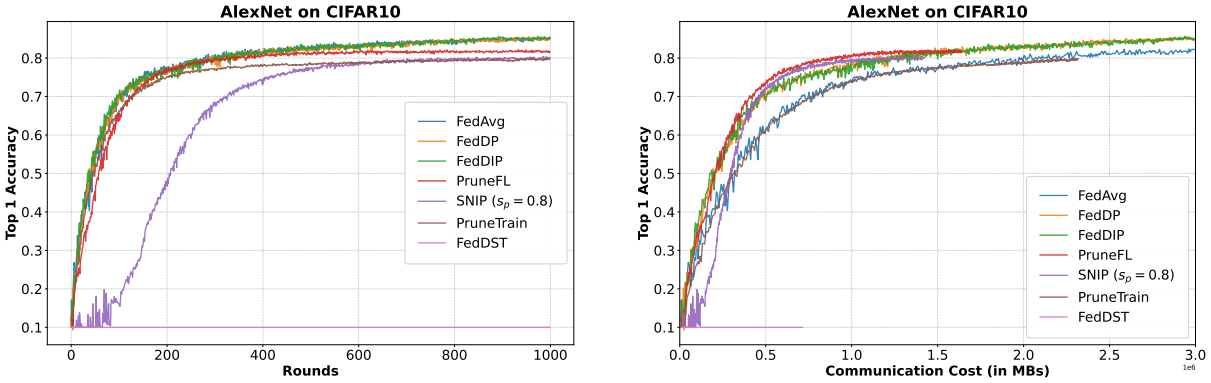
Pruning typically decelerates convergence due to the reduction in informational content available during training. However, Table 5.3 indicates that FedDIP achieves similar communication efficiency to other baselines with minimal performance reduction. For example, in the *Fashion-MNIST* experiment with LeNet-5, FedDIP achieves an accuracy of 86.62% within a communication budget of $4 \cdot 10^3$ MB, outperforming FedAvg and other methods. In the *CIFAR-10* experiment with AlexNet, FedDIP attains 82.58% accuracy within a $1.8 \cdot 10^6$ MB budget. In the *CIFAR-100* experiment with ResNet-18, FedDIP achieves 69.57% accuracy within a $2 \cdot 10^6$ MB budget, balancing performance and communication cost effectively.

Sparse training, utilizing dynamic model pruning with error feedback, significantly reduces computational costs from approximately 50% initially to 90% at the target. While FedDIP incurs minimal computational overhead due to incremental regularization, this overhead is negligible when considering the superior accuracy achieved compared to other pruning baselines. The sizes of the pruned models at target sparsity levels are detailed in Table 5.2.



(a) Test accuracy. (b) Test accuracy vs. communication budget.

Figure 5.5: Fashion-MNIST experiment with LeNet-5.

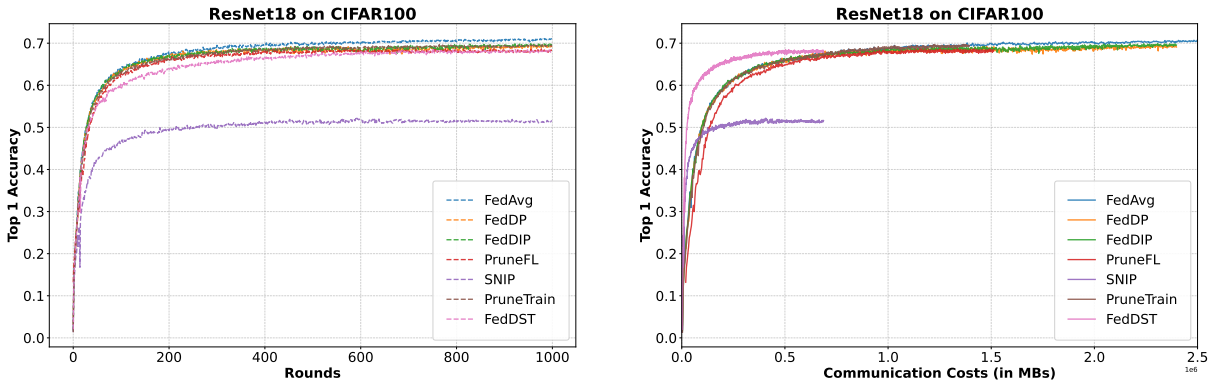


(a) Test accuracy. (b) Test accuracy vs. communication budget.

Figure 5.6: CIFAR-10 experiment with AlexNet.

Experiments with Non-IID Data

Table 5.4 highlights FedDIP's robustness in handling non-i.i.d scenarios effectively, even under extreme sparsity. It shows that FedDIP adapts well to FL frameworks that address non-i.i.d challenges, such as FedProx, without significant performance loss on non-i.i.d data. Compared with FedAvg, the proposed method retains comparable performance even with a 10× compression rate. Especially at $s_p = 0.8$ in *ResNet-18*, FedDIP achieves the highest top-1 accuracy,



(a) Test accuracy.

(b) Test accuracy vs. communication budget.

Figure 5.7: CIFAR-100 experiment with ResNet-18.

Table 5.2: Test Accuracy (Top-1)

	Model Performance (%) with target sparsity s_p		
Model	*LeNet*; $s_p = .9$	*AlexNet*; $s_p = .9$	*ResNet*; $s_p = .8$
FedAvg	89.50 (0.09)	85.07 (0.13)	70.92 (0.10)
FedDP	88.06 (0.08)	84.81 (0.18)	69.23 (0.14)
FedDIP	**88.26** (0.09)	**85.14** (0.22)	**69.67** (0.10)
PruneFL	86.00 (0.10)	81.64 (0.17)	68.17 (0.20)
SNIP	86.08 (0.15)	80.10 (0.15)	51.46 (0.11)
PruneTrain	84.36 (0.10)	79.73 (0.10)	69.39 (0.08)
FedDST	80.37 (0.20)	–	68.06 (0.20)
# param. (FedAvg)	62K	23.3M	11.2M
# param. (pruned)	**6.1K**	**2.3M**	**2.2M**

[1] Mean accuracy; standard deviation in '()'.

Table 5.3: Communication Efficiency

	Model Performance (%) with communication budget		
Case	*LeNet-5*$^{(1)}$	*AlexNet*$^{(2)}$	*ResNet-18*$^{(3)}$
FedAvg	86.76	78.98	70.06
FedDP	86.54	82.29	69.10
FedDIP	**86.62**	**82.58**	**69.57**
PruneFL	85.57	81.73	68.4
SNIP	86.32	80.11	51.63
PruneTrain	82.68	78.16	69.42
FedDST	80.37	–	68.06

Communication budget $^{(1)}4 \cdot 10^3$MB, $^{(2)}1.8 \cdot 10^6$MB, $^{(3)}2 \cdot 10^6$MB

5.6.2 Sparsity Analysis

Layerwise Sparsity

Under different target overall sparsity for ResNet-18 ($s_p = 0.8$), LeNet-5 ($s_p = 0.9$), and AlexNet ($s_p = 0.9$), Figure 5.8 shows the sparsity *per* layer.

Table 5.4: Extension to Non-IID Data

	Model Performance (%) (Non-IID Case)		
Case	*LeNet-5*	*AlexNet*	*ResNet-18*
FedAvg	76.42 (0.28)	61.59 (0.73)	16.44 (0.49)
FedProx	**76.63** (0.34)	**65.74** (0.26)	18.48 (0.91)
FedDIP+Prox	74.49 (0.09)	60.47 (0.52)	**19.22** (0.8)

[1] Mean of the highest five *top-1* test accuracy during T rounds.

Because of the function of general feature extraction, the first layers of all models are the least pruned ($0.1 \leq s \leq 0.4$).

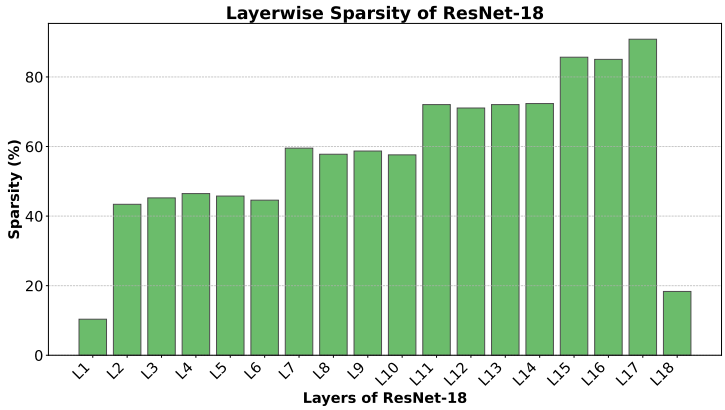
Furthermore, the initialization of sparsity distribution matters. According to Figure 5.8, the dependency of layerwise sparsity on the number of weights per layer relies on the ERK distribution in FedDIP's initialization, despite global magnitude pruning in a later process. The correlation between the number of weights per layer and the corresponding sparsity level stems from the initial ERK distribution, which allocates a higher degree of sparsity to layers containing more weights. Such correlation is remarkable in both convolutional and fully-connected layers of the models. In convolutional layers, the correlations are found to be perfectly linear for *LeNet-5* with a correlation coefficient $\varrho \simeq 1$, for *AlexNet* we obtain $\varrho = 0.86$, while for *ResNet-18* $\varrho = 0.8$. For fully-connected layers, since only one exists in *ResNet-18*, we obtain $\varrho = (0.91, 0.82)$ for *LeNet-5* and *AlexNet*, respectively.

FedDIP in Extreme Sparsity

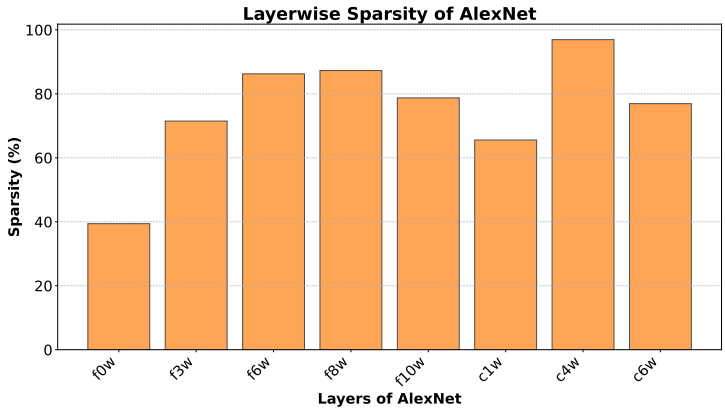
The efficiency of FedDIP is examined under more extreme sparsity. For *Fashion-MNIST* and *CIFAR10* experiments, we investigate two additional extreme sparsity levels $s_p = 0.95$ and $s_p = 0.99$, and for *CIFAR100* experiments, we investigate $s_p = 0.9$ and $s_p = 0.95$. These conditions provide a robust assessment of FedDIP's performance across a range of extreme sparsity. As shown in Figure 5.9, under extreme sparsity like 0.95 and 0.99, the largest drops Δ in classification accuracy are only $\Delta = 6.97\%$, $\Delta = 5.03\%$, and $\Delta = 8.08\%$, respectively. This also comes with *further* $100\times$, $100\times$, and $20\times$ compression rate on LeNet-5, Alex-Net, and ResNet-18 model sizes, respectively. This indicates (i) FedDIP's efficiency in storing and managing models as well as (ii) in inference tasks for subsequent low memory devices. All in all, the pruned DNN models' performance is relatively high with small accuracy drops and high model compression across different tasks.

5.7 Limitations and Future Research

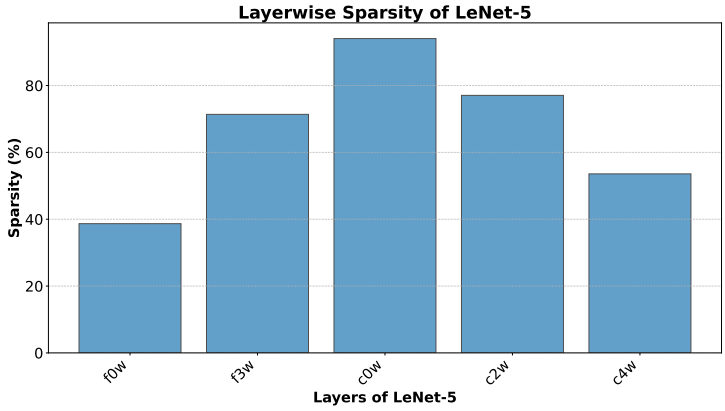
Two primary limitations request attention and investigation in future research.



(a) Distribution of layer sparsity; ResNet-18.



(b) Distribution of layer sparsity; AlexNet.



(c) Distribution of layer sparsity; LeNet-5.

Figure 5.8: Layerwise pruning sparsity; *f0w* stands for (*f*)eatures layer, layer index (e.g, 0), and (*w*)eights, respectively. *c* stands for the classifier layer (the same notation is used for other layers). ResNet-18 consists of 18 pruning layers.

First, the current framework is specifically tailored to Convolutional Neural Networks (CNNs) for image classification tasks using unstructured pruning. However, it should be noted that unstructured pruning often requires high sparsity levels and specific methodologies to achieve performance gains, which may not be feasible or efficient on small NPUs or other embedded platforms. This limitation highlights that current unstructured

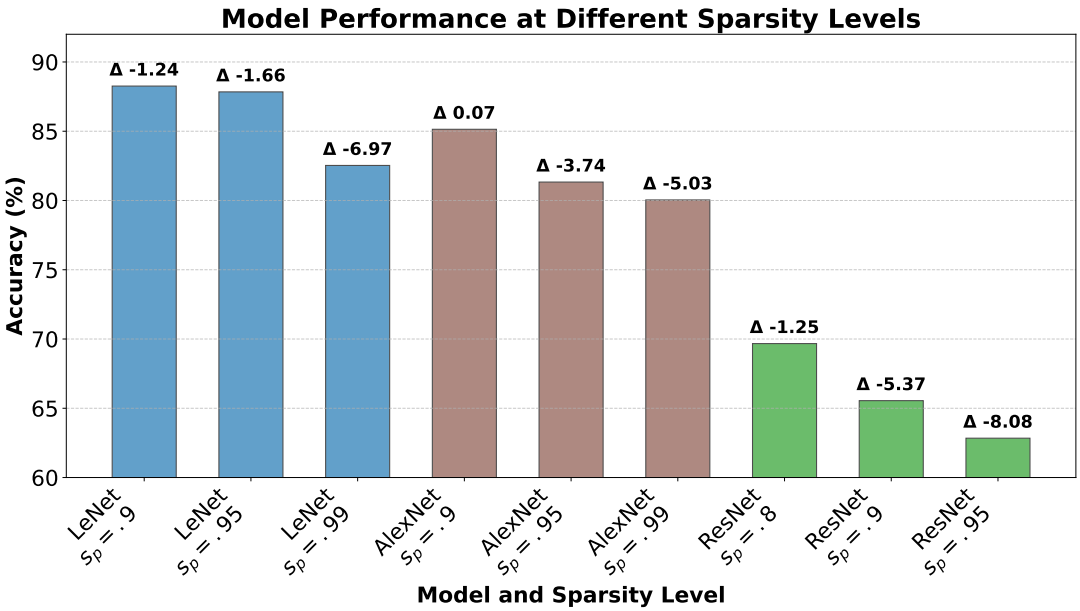


Figure 5.9: FedDIP performance on extreme sparsity values.

pruning approaches, including what we implemented, are not directly applicable to such devices without further adaptation. Additionally, the efficacy of FedDIP in other types of neural networks, such as Recurrent Neural Networks (RNNs) or Transformer models, remains unexplored, thus limiting the generalizability of the proposed approach. Expanding the application of FedDIP to include RNNs and Transformer models could significantly broaden its scope. Moreover, incorporating structured pruning methods, which are better suited for many embedded devices, could provide more practical and efficient solutions for resource-constrained platforms.

Second, although the proposed FedDIP framework addresses dynamic pruning and incremental regularization, its adaptation to algorithms like FedProx for handling heterogeneous data distributions shows potential but does not enhance performance. Non-i.i.d. data distributions can lead to uneven model updates and slower convergence rates, impacting overall model efficacy. Future research should focus on extending the FedDIP framework to effectively manage highly non-i.i.d. data distributions, potentially through model personalization. Developing adaptive mechanisms that dynamically adjust pruning and regularization parameters based on data heterogeneity could improve the framework's performance.

5.8 Conclusions

In this chapter, we have presented FedDIP, an innovative Federated Learning (FL) framework that integrates dynamic pruning and incremental regularization to enhance the efficiency and scalability of deep neural networks in decentralized environments. Our ap-

proach leverages the ERK distribution for data-free initialization, ensuring a balanced sparsity distribution across different layers of the network. Through comprehensive theoretical analysis and extensive empirical evaluations, FedDIP has demonstrated its capability to achieve significant reductions in computational and communication costs while maintaining high model accuracy.

The results from our experiments on benchmark datasets, such as Fashion-MNIST, CIFAR-10, and CIFAR-100, validate the effectiveness of FedDIP in both i.i.d. and non-i.i.d. scenarios. FedDIP consistently outperforms traditional federated learning methods and state-of-the-art pruning techniques, achieving superior accuracy even under extreme sparsity conditions. The layerwise sparsity analysis further highlights the importance of the initial sparsity distribution and its impact on model performance.

By reducing the size of neural network models during the federated learning process, FedDIP not only facilitates a similar convergence rate but also lowers the communication overhead, making federated learning more practical for real-world applications. Additionally, FedDIP's ability to handle extreme sparsity levels underscores its potential for deployment in resource-constrained environments. Hence, this chapter verifies that pruning optimizes the efficiency of distributed machine learning, particularly in federated learning.

The next Chapter will focus on addressing the heterogeneity in personalized and decentralized FL environments and exploring the integration of other model compression techniques to further enhance the efficiency and scalability of federated learning systems. The promising results of FedDIP open new avenues for efficient and scalable model training in decentralized settings.

Chapter 6

Efficient Decentralized Federated Learning with Pruning

6.1 Introduction

In Chapter 5, we substantiated the hypothesis that efficiency in distributed deep learning can be achieved through dynamic extreme pruning. The impressive performance of this pruning method in centralized federated learning (CFL)—where coordination with a central server is essential—motivates us to explore its potential to enhance efficiency within a more robust scenario: decentralized federated learning (DFL). The comparison between CFL and DFL is summarised in Table 6.1 and Figure 6.1

Decentralized Federated Learning (DFL), as introduced by Lalitha et al. (2018), operates by enabling communication and model sharing without the management of central servers. Compared to CFL, DFL is more diverse and flexible, allowing information exchange among clients under various topologies, such as *ring* and *fully-connected*, which will be elaborated on in Section 6.3.1. Additionally, DFL is resilient to central server failures, attacks, and drop-out issues. Specifically, most DFL paradigms continue to function even if a proportion of nodes, including servers, lose connections. However, as noted by Yuan et al. (2024), DFL faces extreme challenges, including high communication volume per round, computational and storage burdens, cybersecurity vulnerabilities, and the absence of incentive mechanisms. For example, following the notation in Chapter 5, we denote the number of parameters to send as $|\omega|$ and the total number of clients as M. The total communication volume for CFL per round is $M|\omega|$, but for DFL, it is $M^2|\omega|$ if the fully-connected topology is considered. Due to the flexibility of DFL, it is often used for cross-device FL, which is a more resource-constrained scenario than CFL. This chapter primarily addresses the first two challenges. Moreover, data heterogeneity remains a significant issue within the learning framework.

The aforementioned challenges raise the question: *Can we efficiently learn under the*

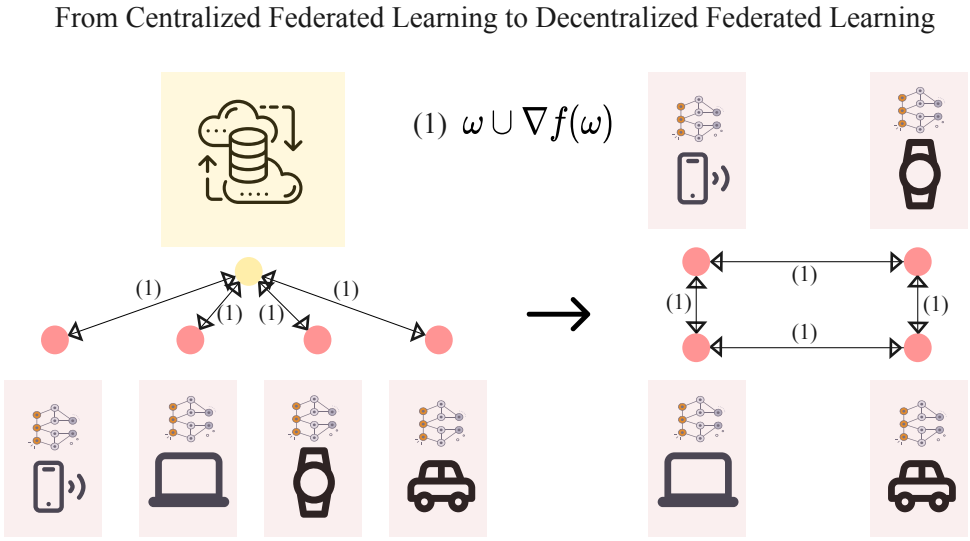


Figure 6.1: **Transition from CFL to DFL: An Example** The model parameters or gradients are exchanged from clients to the central cloud in CFL; In DFL, the information is exchanged among clients without the management of a central server, where the clients are connected with a ring topology.

decentralized federated learning framework while simultaneously enhancing model performance using compression techniques such as pruning? Our answer is an emphatic *yes*. We propose the Dynamic Aggregation Decentralized Personalized Federated Learning (DA-DPFL) framework, which incorporates a dynamic *sparse* to *sparser* training procedure. Across various tasks and degrees of data heterogeneity, DA-DPFL achieves superior energy efficiency (i.e., communication and computation efficiency) compared to state-of-the-art methods. This chapter contributes the following:

- A dynamic aggregation framework that reuses previous clients' models within the same communication round.
- An advanced pruning strategy that prunes models based on their *compressibility*, effectively extending existing pruning methods in DFL.
- Comprehensive experiments demonstrating the learning efficiency of DA-DPFL compared to both CFL and DFL baselines across various tasks, data heterogeneity levels, and model architectures.
- Achievement of the target model's highest energy efficiency and minimal storage requirements.

- A theoretical convergence analysis that supports the proposed learning framework, consistent with experimental findings.

Aspect	Centralized Federated Learning (CFL)	Decentralized Federated Learning (DFL)
Communication	Clients communicate directly with the central server.	Clients communicate directly with each other in certain protocols.
Robustness	Prone to the single point of failure due to dependence on a central server.	More robust as there is no central server
Scalability	Face scalability issues as the number of clients increases.	High scalability due to distributed nature, but increases communication overhead.
Security and privacy	Central servers can be targets for attacks,	Improved privacy as data is distributed and resilient against central attacks.

Table 6.1: Comparison between Centralized Federated Learning (CFL) and Decentralized Federated Learning (DFL).

6.2 Related Work

This section details existing work on leveraging personalized Federated Learning (FL) to address non-i.i.d. problems, the evolution of decentralized FL frameworks, and the utilization of compression techniques to achieve efficiency in FL. Each approach has its own advantages and challenges.

6.2.1 Personalized Federated Learning

The data distribution across different clients and organizations is inherently heterogeneous, posing a fundamental challenge for federated learning (FL). Personalized FL is the technique that aims to tailor the global model or directly learn a personalized model to fit local data distributions (Tan et al., 2023) for individual clients, i.e., dealing with data heterogeneity issues. Li et al. (2020d) underscored the importance of personalization by proposing FedProx, a generalization, and re-parametrization of FedAvg (McMahan et al., 2017). Ditto (Li et al., 2021b) offered a fair personalization framework through globally regularized multitask FL, while FOMO (Zhang et al., 2021b) focused on first-order optimization for personalized learning. FedABC (Wang et al., 2023) employed a 'one-vs-all'

strategy and binary classification loss to address class imbalance and unfair competition. Conversely, FedSLR (Huang et al., 2022b) integrated low-rank global knowledge for efficient communication. However, these methods often increase training costs, highlighting the need for more efficient training approaches. Wang et al. (2024) introduced the pFedHR framework, which tackles model heterogeneity in FL by leveraging heterogeneous model reassembly to generate personalized models for clients dynamically and automatically with minimal human intervention. Recent research suggested that generating personalized masks through pruning can enhance communication efficiency and manage data heterogeneity. Techniques such as FedMask (Li et al., 2021a), FedSpa (Huang et al., 2022a), and DisPFL (Dai et al., 2022) achieved this using personalized masks.

6.2.2 Decentralized Federated Learning

Decentralized federated learning (DFL) has emerged as a robust paradigm for distributed learning, enabling clients to collaboratively train models with their neighbors, thereby enhancing privacy and reducing reliance on central servers (Beltrán et al., 2023). Increased client interaction in DFL has led to methods like DFedAvgM (Sun et al., 2022), which extends FedAvg to decentralized contexts with momentum SGD. Based on the committee mechanism, Che et al. (2022) provided a serverless FL framework, which involves two client selection strategies to be robust to Byzantine attacks. To enhance communication efficiency in DFL, Zhao et al. (2022) designed a *better compression for decentralized optimization* method called BEER, which enhances convergence in non-convex optimization through communication compression and gradient tracking. GossipFL (Tang et al., 2022) is a gossip learning framework that utilizes connection bandwidth information to create a gossip matrix for sparsified gradient communication. Addressing the *non-i.i.d* challenge, DFedSAM (Shi et al., 2023) employed the Sharpness-Aware Minimization (SAM) optimizer, while DisPFL (Dai et al., 2022) leveraged RigL-like pruning for decentralized sparse training with personalization, thus reducing generalization error and communication costs.

6.2.3 Efficient Federated Learning with Pruning

One of the core challenges in this thesis is addressing the significant communication and training costs in distributed machine learning, specifically in federated learning (FL), as discussed in Chapters 5 and 6. Various techniques have been proposed to mitigate these issues. As outlined in Section 5.2, efficient distributed computing through compression has been explored. This section focuses on more recent advancements specifically related to pruning in FL. pFedGate, proposed by Chen et al. (2023), tackled these challenges by adaptively learning sparse local models with a trainable gating layer, enhancing both

model capacity and efficiency. Additionally, Isik et al. (2023) introduced federated probabilistic mask training, coined as FedPM, to optimize model communication with sparse random networks based on a new *Bayesian* aggregation policy. The FedMask (Li et al., 2021a) utilized heterogeneous binary masks to improve communication and computation efficiency in personalized federated learning. Each device learns a sparse, personalized model by applying a binary mask to the fixed parameters of its local model, and only these binary masks are communicated between the server and devices, significantly reducing communication costs while maintaining model performance.

Although significant progress has been made in reducing communication costs, as aforementioned, only a few methods (Jiang et al., 2022; Chen et al., 2023; Li et al., 2021a) have also effectively addressed the reduction of training costs through sparse training.

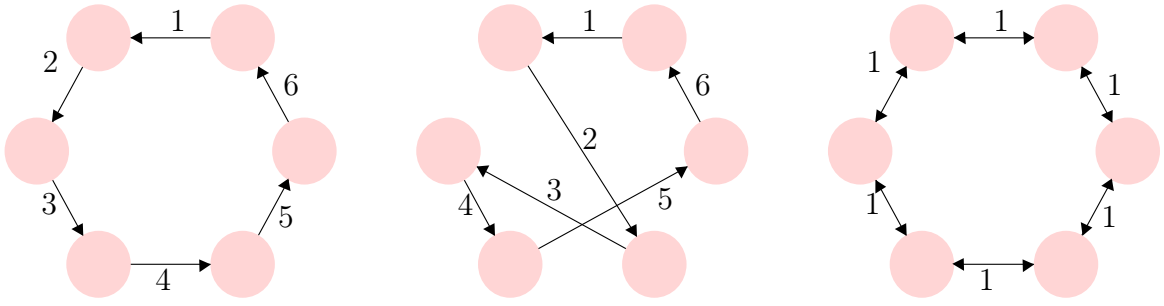
6.3 Methodology

This section first introduces the definition and formulation of decentralized personalized federated learning (DPFL) and then clarifies the rationale and key components in our proposed method DA-DPFL; it concludes the learning framework in detail.

6.3.1 Preliminaries

Taxonomy in Decentralized Federated Learning

In this section, we present the concept and description of taxonomy in DFL, including **iteration order**, **communication protocols**, **network topology**, and **learning paradigms** according to (Beltrán et al., 2023; Yuan et al., 2024). All the notations are included in Table 2.3.



(a) Cycle (b) Random (c)Parallel

Figure 6.2: **Iteration Orders:** Examples (a), (b), and (c) correspond to the *cycle*, *random*, and *parallel* iteration orders, respectively. The number on the line indicates the iteration steps (i.e. rounds), and the arrow denotes the direction of model transmission.

As stated in Section 5.1, centralized FL requires one central server to coordinate clients' training in *parallel*. The random selection, as one type of iteration order in CFL, has little

impact on the convergence of the system. In DFL, the iteration order is critical and can be classified into *parallel*, *semi-parallel*, and *sequential*. Parallel computing in FL means that all clients communicate simultaneously. A related concept is *parallel*, where coordination ensures that participating devices complete local training at the same time. Specifically, the coordinator waits for all updates before proceeding to the next communication round. *Sequential* learning, as used in other computer science areas, involves clients communicating one by one in a certain order, which can be based on additional information such as device capabilities or network conditions. Figure 6.2 illustrates the random, cycle, and parallel iteration orders.

In CFL, the central server *broadcasts* the model to the clients. In DFL, the client can also broadcast models, referred to as the *broadcast protocol*. The *gossip protocol* is well-established in DFL, allowing clients to disseminate and receive models in a one-to-one random strategy. The current popular protocol is the *hybrid* protocol, combining *broadcast* and *gossip* to suit different scenarios.

In CFL, aggregation happens at the central server, which is the primary method to learn from all clients. However, DFL is diverse and flexible due to the different topologies supporting the communication protocol and iteration order. Figure 6.3 illustrates five commonly used topologies, showing client connection status.

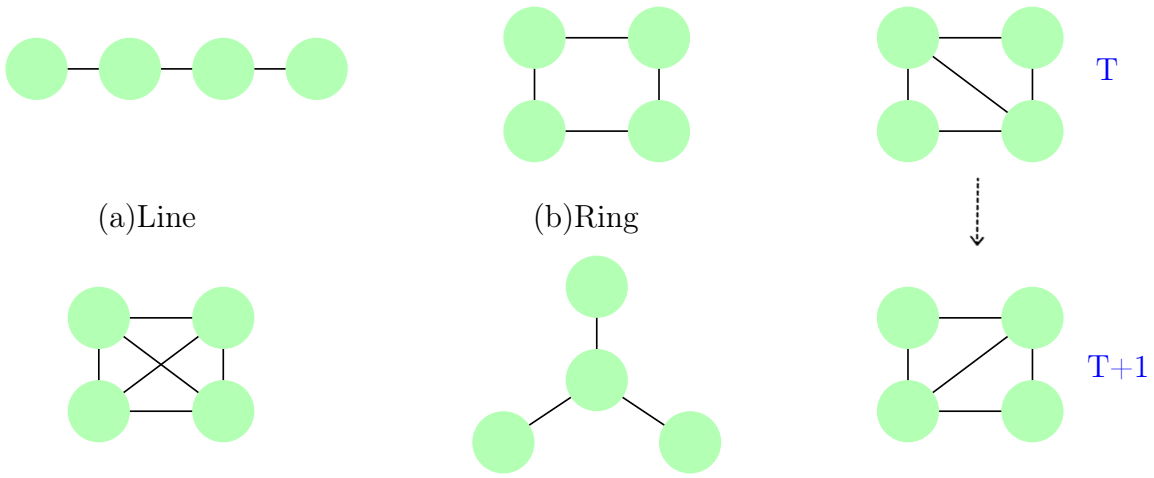
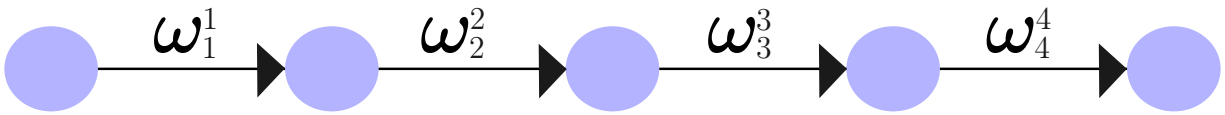


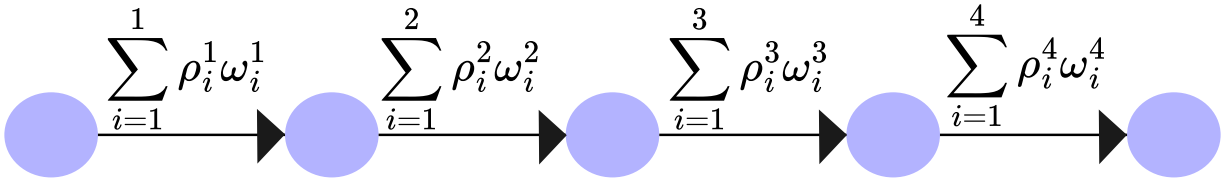
Figure 6.3: **Network Topologies:** Examples of network topologies in Decentralized Federated Learning (DFL). (a) *line* topology, (b) *ring topology*, (c) *fully-connected* (mesh) topology, (d) *star* topology, and (e) *time-varying* topology showing different connections between nodes in subsequent rounds (Round T and Round T+1).

The learning paradigm can be categorized as *aggregate* and *continual* learning, depending on whether the client aggregates the received model. The *aggregate* paradigm follows the CFL aggregation strategy, where received models are aggregated with subsequent lo-

cal training. The *continual* learning, or *incremental* learning, is suitable for DFL and is designed for sequential topologies such as cycles, allowing one client to receive one model from previous clients and perform direct local training without aggregation. Figure 6.4 details the differences, where ω_i^j means the model parameters of client index i at time step j.



(a) Continual Learning



(b) Aggregate Learning

Figure 6.4: **Learning Paradigms:** Illustrations of learning paradigms in Decentralized Federated Learning (DFL). (a) *Continual Learning* where models are updated sequentially, with each client using the model from the previous client. (b) *Aggregate Learning* where models from multiple clients are aggregated at each step. The top-right font denotes the round, and the indices denote the client's model.

DPFL Formulation

As the CFL discussed in Section 5, we introduce a general distributed system with M clients to explain the decentralized, federated learning framework, followed by the formulation of mask-based personalized DFL.

The clients form a network represented by a graph $\mathcal{G}(\mathcal{M}, \mathbf{V})$, where the adjacency matrix $\mathbf{V} = [v_{i,j}] \in \mathbb{R}^{M \times M}$ delineates the communication links between clients. Specifically, client j communicates with client i if $v_{i,j} > 0$, forming the neighborhood $\mathcal{G}_i = \{i \in \mathcal{M} : v_{i,j} > 0\}$. A zero entry $v_{i,j} = 0$ indicates no direct communication. The symmetry $v_{i,j} = v_{j,i}$ does not necessarily hold, implying potential asymmetry in communication.

We focus on dynamic topologies, where the communication structure evolves over time.

At each discrete time step $t \in \mathbb{T} = \{1, 2, \ldots\}$, the adjacency matrix $\mathbf{V}^t$ updates to reflect the current network state. This dynamic nature introduces a time-varying, non-symmetric topology, represented as $\mathbf{V}^t = [v_{i,j}^t] \in \mathbb{R}^{M \times M}$, which defines the temporal neighborhood $\mathcal{G}_i^t$ for each client i.

In our scalable decentralized federated learning (DFL) setup, each client $i \in \mathcal{M}$ has its own local dataset $\mathcal{D}_i = \{(\mathbf{x}, y)\}$, where $\mathbf{x}$ represents input features and y denotes the output labels. Clients communicate with their neighbors $\mathcal{G}_i^t$ to exchange model updates. The goal is to learn personalized models ω_i for each client i by minimizing a global objective:

$$\min_{\{\omega_i\}_{i=1}^M} \frac{1}{M} \sum_{i=1}^{M} F_i(\omega_i), \tag{6.1}$$

where $F_i(\omega_i) = \mathbb{E}[\mathcal{L}(\omega_i; (\mathbf{x}, y)) \mid (\mathbf{x}, y) \in \mathcal{D}_i]$ and $\mathcal{L}(\cdot; \cdot)$ denotes the loss function.

We introduce model pruning to enhance efficiency and realize personalization, where a binary mask $\mathbf{m}$ is applied to each model to eliminate unnecessary weights. This modifies the objective to:

$$\min_{\omega, \{\mathbf{m}_i\}_{i=1}^M} \frac{1}{M} \sum_{i=1}^{M} F_i(\omega \odot \mathbf{m}_i), \tag{6.2}$$

where $F_i(\omega \odot \mathbf{m}_i) = \mathbb{E}[\mathcal{L}(\omega \odot \mathbf{m}_i; (\mathbf{x}, y)) \mid (\mathbf{x}, y) \in \mathcal{D}_i]$and $\odot$ represents the element-wise multiplication. Each mask $\mathbf{m}_i$ is specific to client i and indicates which weights are retained. The sparsity level s_i of mask $\mathbf{m}_i$ quantifies the proportion of non-zero weights.

The objective is to determine a global model ω and individual masks $\mathbf{m}_i$ such that the resulting personalized models $\omega_i = \omega \odot \mathbf{m}_i$ are optimized for each client $m \in \mathcal{M}$. Communication between clients at time t is limited to their immediate neighbors $\mathcal{G}_i^t$ as defined by the dynamic topology $\mathbf{V}^t$.

6.3.2 Dynamic Aggregation with Controlled Delay

This section details the scheduling policy used for client participation in our framework, which accommodates various topologies such as *ring* or *fully-connected* networks. DA-DPFL is particularly effective for dynamic, time-varying topologies but can also adapt to static topologies, represented as $\mathcal{G}_i$. At the beginning of each communication round t, *reuse indexes*, $\mathcal{N}_i^t$, for neighborhood sets are assigned randomly to C clients. Here, $|\mathcal{G}_i^t| = |\mathcal{N}_i^t| = C < M$, and $\mathcal{G}_i^t \overset{\pi_i^t}{\leftrightarrow} \mathcal{N}_i^t$, where π_i^t is a random bijection mapping. For discrete sets $\mathcal{G}_i^t$ and $\mathcal{N}_i^t$,

$$i = \pi_i^t(j), \tag{6.3}$$

with index $i \in \mathcal{N}_i^t$ and $j \in \mathcal{G}_i^t$. It is possible for $\mathcal{N}_i^t$ to match $\mathcal{G}_i^t$ if the sets are randomly generated. For simplicity, we assume $\mathcal{N}_i^t = \mathcal{G}_i^t$ in Fig. 6.5, where client i is identified by reuse index i. The introduction of $\mathcal{N}_i^t$ highlights the independence in generating *reuse* indexes, crucial for guiding the dynamic aggregation process.

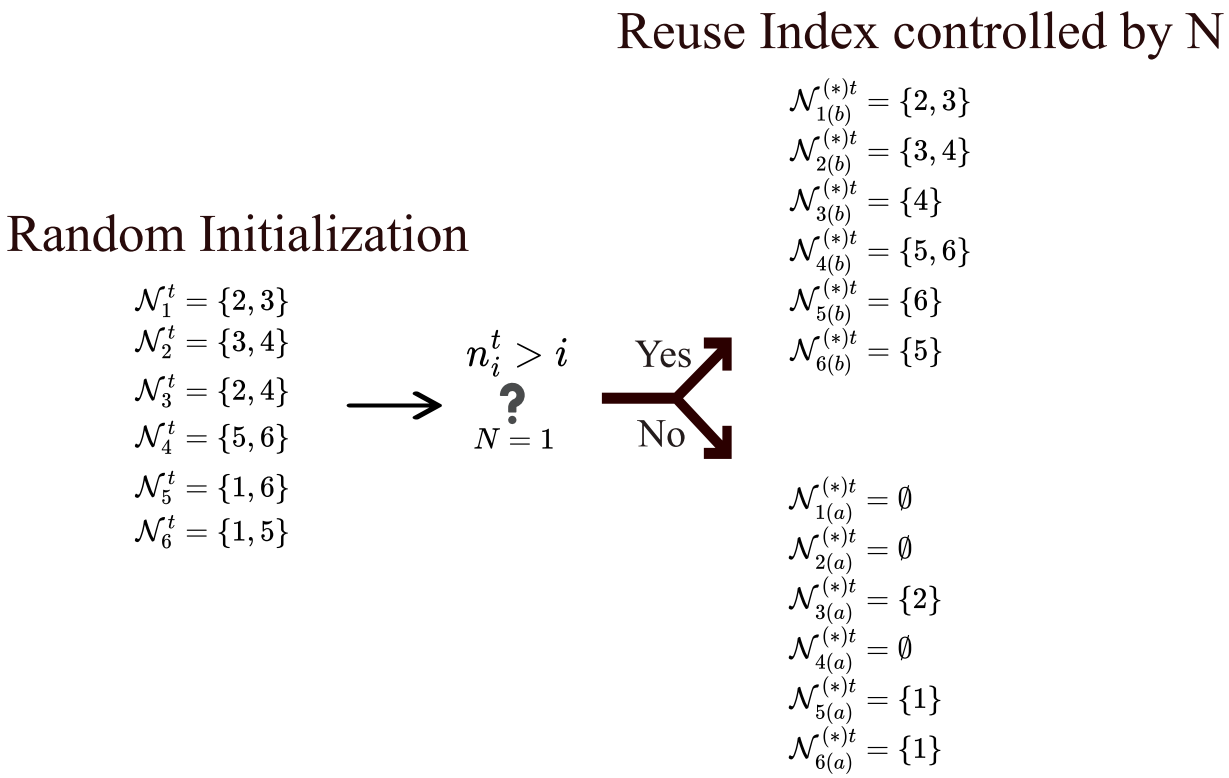


Figure 6.5: **Reuse Index:** Examples of generating reuse indexes with *posterior* and *prior* set under waiting threshold $N = 1$, where $M = 6, C = 2, N = 1$ and assume $\mathcal{N}_i^t = \mathcal{G}_i^t$ for simplicity.

While the criteria for establishing $\mathcal{G}_i^t$ include factors like network bandwidth, geographical location, and link availability, $\mathcal{N}_i^t$ is independent of these factors. Client indices are reassigned for each training round.

In DA-DPFL, a client i might delay receiving models from some neighbors based on $\mathcal{N}_i^t$. We split $\mathcal{N}_i^t$ into two subsets for each client i: (a) the *prior* client subset $\mathcal{N}_{(a)i}^t = \{n_i^t \leq i : n_i^t \in \mathcal{N}_i^t\}$, and (b) the *posterior* client subset $\mathcal{N}_{(b)i}^t = \{n_i^t > i : n_i^t \in \mathcal{N}_i^t\}$. If $\mathcal{N}_{(b)i}^t = \emptyset$, implying $\mathcal{N}_{(a)i}^t = \mathcal{N}_i^t$, client i waits for the **slowest** client within $\mathcal{N}_i^t$ before starting model aggregation and local dataset training $\mathcal{D}_i$. To improve scalability, we introduce a threshold allowing the client to wait for, at most, N of the fastest clients in $\mathcal{N}_{(a)i}^t$, where $|\mathcal{N}_{(a)i}^{(*)t}| = N \leq C$. Then, $\mathcal{N}_{(a)i}^{(*)t} \cup \mathcal{N}_{(b)i}^{(*)t} = \mathcal{N}_i^t$. Conversely, if no prior client set exists ($\mathcal{N}_{(a)i}^{(*)t} = \emptyset$), client i receives models from $\mathcal{N}_{(b)i}^{(*)t}$ without delay, as shown by nodes 1, 2, and 4 in Fig. 6.5. Based on the bijection mapping between $\mathcal{N}_i^t$ and $\mathcal{G}_i^t$, $\mathcal{G}_{(a)i}^{(*)t}$ is determined.

DA-DPFL's learning schedule significantly differs from traditional FL paradigms, im-

plementing a semi-parallel approach. This hybrid scheme combines two iteration orders, *sequential* and *parallel*, as discussed in Section 6.3.1. Specifically, it involves *sequential* learning with delayed aggregation and immediate *parallel* aggregation, known as dynamic aggregation. Continual learning is achieved by sequentially learning models from clients in the prior set of client i, benefiting from sequential knowledge transfer. This may delay client i in aggregating models from $\mathcal{G}_{i(a)}^{(*)t}$. Conversely, models from posterior clients are sent to client i independently, enabling training parallelism. At time t, nodes $\{1, 2, 4\}$ engage in simultaneous (parallel) training, while nodes $\{3, 5, 6\}$ await model reuse from preceding clients. This allows subsequent nodes to train concurrently with earlier ones, as shown by nodes $\{5, 6\}$ training alongside node 3. By introducing a cutoff value N, our waiting policy gains controllability. When $N = 0$, DA-DPFL functions as a parallel FL system with sparse training; when $N = C = M$, it transitions to a *sequential* FL.

6.3.3 Dynamic Pruning

To innovate a pruning strategy, three critical questions must be addressed: *what to prune*, *how to prune*, and *when to prune*. For the first question, similar to the method discussed in Section 5, we employ *unstructured* pruning, which involves pruning model weights on an entry-wise basis. The distinctions between structured and unstructured pruning are detailed in Section 5.3.

Regarding *how* to prune, extensive research has been conducted, as discussed in Sections 5.2and 6.2, Before introducing the main technique employed in DA-DPFL, which is adapted and modified from centralized pruning, we provide a brief overview of a significant hypothesis that motivates pruning. The Lottery Ticket Hypothesis (LTH, Frankle and Carbin (2018)) posits that *winning tickets*, or properly pruned networks, enhance the learning performance of the original unpruned network. Evci et al. (2020) proposed an algorithm called RigL, which identifies all the *winning* tickets. This method dynamically trains sparse neural networks from *scratch*, maintaining a fixed number of model parameters and, consequently, a fixed computational burden, which is essential for federated learning. This is achieved through proper initialization, parameter dropping, and regrowth. RigL involves the following steps:

- Initialize with the *Erdos-Renyi-Kernel* (ERK) distribution.
- Update with a *Cosine* Annealing Schedule.
- Drop the *top-k* weights based on their magnitude.
- Grow the *top-k* connections based on gradient magnitude.

The update schedule follows a decay function f_{decay}, which quantifies the fractions of updated connections. f_{decay} is defined as:

$$f_{decay}(t, \alpha, T) = \frac{\alpha}{2}(1 + \cos\frac{t\pi}{T}) \tag{6.4}$$

Updates occur every R configuration rounds, hence $t \in [0, R, 2R, \ldots, T]$, where T represents the total training rounds in a centralized setup and also the communication rounds in decentralized training. α denotes the initial fraction to be updated. It decays from the initial fraction α to 0 when $\cos\pi = -1$ at $t = T$. The details of the extension for RigL to DA-DPFL are discussed in the next section (Algorithm 11).

While the sparse model training with a fixed computation budget is promising, *is it possible to further reduce the size of the sparse model to achieve greater efficiency?* We propose a *sparse-to-sparser* pruning strategy. We define compressibility as the degree to which neural networks can be reduced without significantly decreasing performance. The PQ-Index used in the Sparsity-informed Adaptive Pruning (SAP) algorithm, proposed by Diao et al. (2022), provides the technique to measure "compressibility". The PQ-Index, i.e., the ratio of prunable weights, is formulated as:

$$\mathcal{I}(w_t) = 1 - d_t^{\frac{1}{q}-\frac{1}{p}} \frac{\|\boldsymbol{\omega}_t\|_p}{\|\boldsymbol{\omega}_t\|_q} \tag{6.5}$$

where d_t is the density of the model and $p < q$ are numerical values. The details of the adapted usage of PQI are clarified in the next section, Algorithm 10.

The final question is *when* to prune. Most research on pruning, such as (Evci et al., 2020; Diao et al., 2022; Ruan et al., 2020; Jiang et al., 2022), updates the pruning mask at constant configuration rounds, i.e., the mask is updated every R training/communication rounds. Considering two extreme cases underscores the importance of pruning timing. If R is sufficiently large, dynamic pruning during training will resemble post-training pruning, which forfeits the advantages of reducing computational overhead in centralized training and both computational and communication overhead in decentralized training. Conversely, if R is sufficiently small and initialization is improper, the short-term learned information cannot sufficiently guide the pruning based on the importance of connections or weights, resulting in model training divergence. Therefore, DA-DPFL modifies the criteria used in EarlyCrop (Rachwan et al., 2022) to decide when to prune further. The pruning time detection score is:

$$\frac{|\Delta_0^t - \Delta_0^{t-1}|}{|\Delta_0^1|} < \delta_{pr}; \Delta_0^t := \|\boldsymbol{\omega}^t - \boldsymbol{\omega}^0\|^2 \tag{6.6}$$

where δ_{pr} is a threshold to detect time to prune.

Algorithm 9 The DA-DPFL Algorithm

Input: M clients; T, E_l rounds; PQI hyper-param. $\{p, q, \gamma, \eta_c\}$, pruning thr δ_{pr}; voting threshold δ_v; factors b, c; target sparsity s^*;
Output: Personalized aggregated models $\{\tilde{\boldsymbol{\omega}}_i^T\}_{i=1}^M$.
Initialization: Initialize $\{\mathbf{m}_i^0\}_{i=1}^M$, $\{\boldsymbol{\omega}_i^0\}_{i=1}^M$, $\mathcal{T} \leftarrow \emptyset$
for round $t = 1$ **to** T **do**
  for each client i **do**
    Generate a random *reuse* index set $\{\mathcal{N}_i^t\}_{i=1}^M$.
    Generate a random bijection π_i^t between $\mathcal{N}_i^t$ and $\mathcal{G}_i^t$
    Form prior and posterior set $\{\mathcal{N}_{i(a)}^{(*)t}, \mathcal{N}_{i(b)}^{(*)t}\}$
    Form $\{\mathcal{G}_{i(a)}^{(*)t}, \mathcal{G}_{i(b)}^{(*)t}\}$ by $\{\mathcal{N}_{i(a)}^{(*)t}, \mathcal{N}_{i(b)}^{(*)t}, \pi_i^{-1(t)}\}$
    if $\mathcal{G}_{(a)i}^{(*)t} \neq \emptyset$ **then**
      do Wait models from neighbors $\mathcal{G}_{(a)i}^{(*)t}$
    end if
    Receive neighbor's models $\boldsymbol{\omega}_j^t, j \in \mathcal{G}_i^t$
    Obtain mask-based aggregated model $\tilde{\boldsymbol{\omega}}_i^t$.
    Compute $\tilde{\boldsymbol{\omega}}_{i,\tau}^t$ for E_l local rounds.
    Calculate $\Delta_0^t(i)$ and $v_t(i)$ based on δ_{pr}.
    Broadcast $v_t(i)$ to all clients; derive t^*.
    if $t \in \mathcal{T}$ **and** $s_i < s^*$ **then**
      Call Algorithm 10 to obtain $\tilde{\boldsymbol{\omega}}_{i,E_l}^{t\prime}, \mathbf{m}_i^{t\prime}$
      Update sparsity s_i
    else
      Set $(\tilde{\boldsymbol{\omega}}_{i,E_l}^{t\prime}, \mathbf{m}_i^{t\prime}) \leftarrow (\tilde{\boldsymbol{\omega}}_{i,E_l}^{t}, \mathbf{m}_i^{t})$
    end if
    Call Algorithm 11 to update $\mathbf{m}_i^{t+1}$
    Set $\boldsymbol{\omega}_i^{t+1} = \tilde{\boldsymbol{\omega}}_{i,E_l}^{t\prime}$
  end for
  if $t == t^*$ **then** Update $\mathcal{T}$.
end for

6.3.4 DA-DPFL Algorithm

According to the aforementioned, the DA-DPFL leverages the dynamic aggregation to reduce the required communication rounds to achieve convergence because of the reuse of models from clients in *prior* set; achieves communication and computation efficiency by further reducing the size of models, which is supported by the innovative pruning strategy for decentralized training. This section details the DA-DPFL algorithm, where further details are explained in Algorithm 9.

The *reuse* index generation to navigate dynamic aggregation has been introduced in Section 6.3.2. To adapt the pruning methods used in centralized training for decentralized federated learning, we give the following clarification about the time-optimized dynamic pruning policy.

A dynamic pruning policy is introduced to complement the scheduling of local training and gradual model aggregation via prior and posterior neighbors for each client. Initially, each client's mask $\mathbf{m}_i^0$ is configured using the Erdos-Renyi Kernel (ERK) distribution Evci et al. (2020). These masks are then dynamically adjusted by pruning and regrowing connections based on importance scores derived from the magnitudes of model weights and gradients. This approach extends the centralized RigL method (Evci et al., 2020) to DA-DPFL, as elaborated Algorithm 11.

The proposed method operates orthogonally to other fixed-sparsity training techniques, such as RigL, enabling further pruning stages. The Sparsity-informed Adaptive Pruning (SAP) algorithm (Diao et al., 2022) introduces the PQ Index (PQI) to measure the "compressibility" of a deep neural network (DNN), as detailed in Algorithm 10. DA-DPFL incorporates PQI within DFL, addressing the heterogeneity of various local models by adaptively pruning each model according to its characteristics.

In centralized learning contexts, EarlyCrop (Rachwan et al., 2022) bases its pruning schedule on critical learning periods, while CriticalFL Yan et al. (2023) emphasizes the early doubling of information transmission. EarlyCrop leverages gradient flow and neural tangent kernel dynamics to ensure a smooth transition into pruning. Specifically, the pruning time detection score is calculated according to Equation (6.6). In DA-DPFL, with M clients, a *voting majority rule* is implemented to determine the pruning times. Each client's vote at time t is defined as:

$$v_t(i) = \begin{cases} 1 & \text{if } \frac{|\Delta_0^t(i) - \Delta_0^{t-1}(i)|}{|\Delta_0^1(i)|} < \delta_{pr}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.7}$$

The initial pruning time t^* is then determined by:

$$t^* = \min \left\{ t : \frac{1}{M} \sum_{i=1}^{M} v_t(i) < \delta_v \right\}, \tag{6.8}$$

where δ_v is the voting threshold.

Following the determination of the initial pruning time t^*, DA-DPFL sets the frequency of subsequent pruning events. This strategy is informed by the early training phase, known as the *critical learning period* (Jastrzebski et al., 2021), which impacts the local curvature of the DNN's loss function. During the initial stages, a lower pruning frequency is permitted, which intensifies as the model converges. This balance minimizes communication overhead while maintaining model performance, resulting in an optimal pruning frequency that adapts to different tasks and model architectures.

The *pruning frequency*, or the interval between pruning events, is defined by non-

uniformly dividing the remaining training horizon $T - t^*$:

$$I_\tau := \left\lceil \frac{t^* + b}{c^{\tau-1}} \right\rceil, \quad \tau \in \mathbb{Z}_{\geq 1}, \tag{6.9}$$

where parameter $b > 0$ delays the initial pruning time, and $c > 0$ is a scaling factor adjusting the pruning frequency. The p-th pruning time t_p, for $t_p > t^*$, is calculated as $t_p = \sum_{\tau=1}^{p} I_\tau$, resulting in the set of pruning times $\mathcal{T} = \{t_1, \ldots, t_p : t^* < t_p < T\}$.

Since personalization is achieved through each client's personalized mask for the global model, a mask-based model aggregation policy is appropriate, even though the masks may be obtained through different methods. Therefore, the aggregation policy from DisPFL Dai et al. (2022) and FedDST Bibikar et al. (2022) is adopted. The aggregated model $\tilde{\boldsymbol{\omega}}_i^t$ for client i is derived from the models of its neighbors in $\mathcal{G}_i^t$ at round t using a mask-based approach:

$$\tilde{\boldsymbol{\omega}}_i^t = \left(\frac{\sum_{j \in \mathcal{G}_{i+}^t} \boldsymbol{\omega}_j^t}{\sum_{j \in \mathcal{G}_{i+}^t} \mathbf{m}_j^t} \right) \odot \mathbf{m}_i^t, \tag{6.10}$$

where $\mathcal{G}_{i+}^t = \mathcal{G}_i^t \cup \{m\}$ represents the neighborhood of client i including itself. The local training rounds, denoted by $\tau \in E_l$, based on the obtained $\tilde{\boldsymbol{\omega}}_i^t$ are defined as follows:

$$\tilde{\boldsymbol{\omega}}_{i,\tau+1}^t = \tilde{\boldsymbol{\omega}}_{i,\tau}^t - \eta(\mathbf{g}_{i,\tau}^t \odot \mathbf{m}_i^t), \tag{6.11}$$

where $\mathbf{g}_{i,\tau}^t$ is the gradient of the local loss function $F_i(\cdot)$ with respect to $\tilde{\boldsymbol{\omega}}_{i,\tau}^t$.

Algorithm 10 PQI-driven pruning (Layerwise)

Input: $\tilde{\boldsymbol{\omega}}_{i,E_l}^t$, mask $\mathbf{m}_i^t$, norm index $0 < p \leq 1 < q$, compression hyper-parameter η_c, scaling factor γ, pruning threshold β, further pruning time $\mathcal{T}$.
Output: $\tilde{\boldsymbol{\omega}}_{i,E_l}^{t\prime}$, corresponding mask $\mathbf{m}_i^{t\prime}$
for $t \in \mathcal{T}$ **do**
  for each layer $l \in |L|$ **do**
    Compute dimensionality of $\tilde{\boldsymbol{\omega}}_{i,E_l}^{l,t}$: $d_t^l = |\mathbf{m}_i^{l,t}|$
    Compute PQ Index $I(\tilde{\boldsymbol{\omega}}_{i,E_l}^{l,t}) = 1 - \left(\frac{1}{d_t^l}\right)^{\frac{1}{q}-\frac{1}{p}} \frac{\|\tilde{\boldsymbol{\omega}}_{i,E_l}^{l,t}\|_p}{\|\tilde{\boldsymbol{\omega}}_{i,E_l}^{l,t}\|_q}$
    Compute the lower boundary required model parameters to keep $r_t^l = d_t^l(1 + \eta_c)^{-\frac{q}{q-p}} \left[1 - I(\tilde{\boldsymbol{\omega}}_{i,E_l}^{l,t})\right]^{\frac{p}{q-p}}$
    Compute the number of model parameters to prune
    $c_t^l = \left\lfloor d_t^l \cdot \min\left(\gamma\left(1 - \frac{r_t^l}{d_t^l}\right), \beta\right) \right\rfloor$
    Prune c_t^l model parameters with the smallest magnitude based on $\tilde{\boldsymbol{\omega}}_{i,E_l}^{l,t}$ and $\mathbf{m}_i^{l,t}$
    Find new layer mask $\mathbf{m}_i^{l,t\prime}$ and pruned model $\tilde{\boldsymbol{\omega}}_{i,E_l}^{l,t\prime}$ at layer l
  end for
  Obtain $\tilde{\boldsymbol{\omega}}_{i,E_l}^{t\prime}$ and corresponding mask $\mathbf{m}_i^{t\prime}$
end for

Algorithm 11 RigL mask generation

1: **Input:** $\tilde{\boldsymbol{\omega}}_{i,E_l}^{t\prime}$, corresponding mask $\mathbf{m}_i^{t\prime}$, global rounds T, initial annealing ratio α_0
2: **Output:** New mask $\mathbf{m}_i^{t+1}$
3: Compute prune ratio $\alpha_t = \frac{\alpha}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)$
4: Sample one batch of local training data to calculate dense gradient $\mathbf{g}(\tilde{\boldsymbol{\omega}}_{i,E_l}^{t\prime})$
5: **for** each layer $l \in |L|$ **do**
6: Update mask $\mathbf{m}_i^{l,t+\frac{1}{2}\prime}$ by pruning α_t percentage of weights based on weight magnitude.
7: Update mask $\mathbf{m}_i^{l,t+1}$ via regrowing weights with gradient information $\mathbf{g}(\tilde{\boldsymbol{\omega}}_{i,E_l}^{t\prime})$.
8: **end for**
9: Find new mask $\mathbf{m}_i^{t+1}$.

Note that we exclusively utilize the PQ index to assess compressibility. The differences between the SAP algorithm proposed by Diao et al. (2022) and our method are as follows. First, the SAP algorithm begins with a dense mask, whereas our models are initially sparse across different clients. Second, our proposed strategy functions as an further pruning technique within the federated learning (FL) framework. Third, we emphasize the importance of pruning time, particularly the critical learning period, in contrast to the regular pruning frequency used in the SAP algorithm. Finally, our method implements dynamic pruning during training rather than post-training pruning.

6.4 Theoretical Analysis

Section 5.4 gives the theory support for the pruning with error feedback in the context of CFL, where the quality of pruning is defined by δ_t in Equation (5.23). This section adopts a different pruning method and leverages mask for global model personalization. Then such a quality of pruning can be regarded as the discrepancy between local and global models since different clients hold different masks, as in Equation (6.2). Hence, specific assumptions to quantify the variance for the global and personalized global gradients are provided as in Assumption 5 and 6.

Assumption 5. ***Bounded variance for gradients:*** *Follow (Sun et al., 2022),* $\forall m \in |M|$ *and* $\boldsymbol{\omega} \in \mathbb{R}^d$*:*

$$\mathbb{E}[\|\nabla \hat{f}_i(\boldsymbol{\omega}) - \nabla f_i(\boldsymbol{\omega})\|^2] \leq \sigma_l^2, \tag{6.12}$$

$$\frac{1}{M}\sum_{k=1}^{M} \|\nabla f_i(\boldsymbol{\omega}) - \nabla f(\boldsymbol{\omega})\|^2 \leq \sigma_g^2, \tag{6.13}$$

$$\frac{1}{M}\sum_{k=1}^{M} \|\nabla \tilde{f}_i(\boldsymbol{\omega} \odot \mathbf{m}_i) - \nabla f(\boldsymbol{\omega})\|^2 \leq \sigma_p^2, \tag{6.14}$$

$\hat{f}(\cdot)$ is the estimated gradients from training data; $\tilde{f}(\cdot)$ is personalized global gradients.

Assumption 6. *The aggregated model $\tilde{\boldsymbol{\omega}}_k^t$ for client k at iteration t is given by:*

$$\tilde{\boldsymbol{\omega}}_i^t = \left(\frac{\sum_{j\in\mathcal{G}_i^t}\boldsymbol{\omega}_j^t}{\sum_{j\in\mathcal{G}_i^t}\mathbf{m}_j^t}\right)\odot\mathbf{m}_i^t = \left(\frac{\sum_{j\in\mathcal{G}_i^t}\boldsymbol{\omega}_j^t}{C}\right)\odot\mathbf{m}_i^t \tag{6.15}$$

where $\mathcal{G}_i^t$ is neighborhood of client m with size $|\mathcal{G}_i^t| = C$; all local models are sparse, i.e., $\boldsymbol{\omega}_j^t = \boldsymbol{\omega}_j^t \odot \mathbf{m}_j^t$.

The following theorem supports the convergence of DA-DPFL under the time-varying connected topology.

Theorem 3. *Under Assumptions 2, 5 and 6, when T is sufficiently large and the stepsize η for SGD for training client models satisfies $\eta \leq \sqrt{\frac{1}{12\mu^2(C-1)(2C-1)}}$ for $C > 1$,*

$$\min\mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}^t)\|^2 \leq \frac{2}{T\left(\eta - 6S_1(\mu-\eta)\right)}\left(\mathbb{E}[f(\tilde{\boldsymbol{\omega}}^0)] - \min f\right) + S_3, \tag{6.16}$$

where $S_1 = 2\eta^2C(C-1)\left(\exp\left(\frac{(3C+2)E_l}{4(C^2-1)}\right)-1\right)$, $S_2 = \frac{1}{2C-1}\sigma_l^2 + 3(2\sigma_g^2+\sigma_p^2)$, and $S_3 = \frac{2}{\eta-6S_1(\mu-\eta)}\cdot\left[(\mu-\eta)S_1S_2 + \frac{3\mu^2\eta^3(3C+2)E_l}{2(C+1)C}(\sigma_l^2+\sigma_g^2)\right]$. $f(\tilde{\boldsymbol{\omega}}^0)$ represents the initial global model loss, $\min f$ is the minimum of loss, and C is the neighborhood size.

Proof. See details in Appendix A.4 □

Findings from Theorem 3 indicate that, with sufficiently large T, the error due to initial model loss and bounded variance for gradients becomes negligible. Specifically, choosing $\eta = \mathcal{O}(\frac{1}{\mu\sqrt{T}})$ results in the convergence boundary being dominated by the rate of $\mathcal{O}\left(\frac{1}{\sqrt{T}} + \frac{\sigma_l^2+\sigma_g^2+\sigma_p^2}{\sqrt{T}} + \frac{\sigma_l^2+\sigma_g^2}{T}\right)$.

Additionally, Theorem 3 aligns with two key empirical observations: (i) The number of communication rounds required to achieve a specified error level ε is lower compared to the DisPFL model. This efficiency gain is attributed to the term $S_1 > \left(e^{\frac{E_l}{(2C-2)}}-1\right)$, indicating that no scheduling is involved, which arises from the proposed scheduling strategy. Dividing the left-hand side (first and third item) of the inequality by S_1 results in a reduced error boundary. (ii) The modified ratio is $\frac{3C+2}{2C+2}$. When $C = 2$, the ratio simplifies to $\frac{4}{3}$. As C increases, this ratio approaches $\frac{3}{2}$. This suggests that while increasing C enhances error-bound reduction, the improvement rate diminishes, indicating a limit to the benefits offered by DA-DPFL scheduling.

6.5 Experiment Setup

6.5.1 Datasets and Models

In addition to the CIFAR10 and CIFAR100 datasets used in Chapter 5 (refer to Section 5.5), experiments were also conducted on the medical image dataset HAM10000 (Tschandl et al., 2018). Detailed information about this dataset is provided at the end of this section. The feasibility of the proposed method was verified through experiments on different model architectures, specifically AlexNet (Krizhevsky et al., 2012), ResNet18 (He et al., 2016), and VGG11 (Simonyan and Zisserman, 2015), tested on HAM10000, CIFAR10, and CIFAR100, respectively.

Recall the challenges mentioned in Section 2; data heterogeneity in federated learning (FL) is a significant challenge. To address this, two widely used data partition methods from (Dai et al., 2022) were employed to simulate the non-i.i.d. scenario. The first method, *Dirichlet partition*, uses a Dirichlet distribution with a hyper-parameter α to control the degree of heterogeneity in the local data's label distribution. A larger α results in a more even distribution of data. For example, $\alpha = 100$ approximates an i.i.d. partition, while $\alpha = 0.1$ represents an extreme non-i.i.d. case. The second method, *Pathological partition*, controls the label distribution by defining the exact number of classes (n_{cls}) held by each client. For example, in the CIFAR10 dataset, if $n_{cls} = 2$, some clients have only 2 out of 10 classes. Experiments were conducted with $\alpha = 0.3$ for CIFAR10 and CIFAR100, and $\alpha = 0.5$ for HAM10000. For Pathological partitioning, n_{cls} values were set as follows: 2 for CIFAR10 and HAM10000, and 10 for CIFAR100.

Note: HAM10000 consists of 10,015 dermatoscopic images intended to assist in diagnosing pigmented skin lesions. Each image has a resolution of 600 x 450 pixels and covers a wide range of diagnostic categories related to pigmented lesions. The dataset, sourced from diverse populations using different modalities, is characterized by its comprehensive representation and notable class imbalance. To address this imbalance, an oversampling data augmentation technique was applied, enhancing the representation of minority classes within a training set of 8,000 samples. This approach ensured that each category was approximately equalized to around 6,000 images. Specifically, the minority classes in the randomly split training set (comprising 80% of the data) were augmented, as shown in Table 6.2. This augmentation involved increasing the size of minority classes to integral multiples of their original count, aligning them with the training size of the majority class.

6.5.2 Baselines

The proposed DA-DPFL algorithm is compared with various state-of-the-art baseline algorithms from Centralized Federated Learning (CFL), Personalized Federated Learning

Lesion Type	Original Train Size	Augmented Train Size
Actinic keratoses	297	5346
Basal cell carcinoma	479	5748
Benign keratosis-like lesions	1011	6066
Dermatofibroma	107	5671
Melanocytic nevi	5822	5822
Vascular lesions	129	5676
Melanoma	1067	6402

Table 6.2: Lesion types and train lengths.

(PFL), and Decentralized Federated Learning (DFL). The details are summarized below:

Algorithm	Comp.	Comm.	Heter.
FedAvg	×	×	×
Ditto	×	×	✓
FedDST	✓	✓	×
GossipFL	×	✓	×
DFedAvgM	×	✓	×
DisPFL	✓	✓	✓
BEER	×	✓	×
DFedSAM	×	×	✓
DA-DPFL (ours)	✓	✓	✓

Table 6.3: Comparison of Baseline Algorithms. Comp. is the abbreviation for Computational Efficiency; Comm. is the abbreviation for Communication Efficiency; and Heter. denotes whether the method handles data heterogeneity.

- **FedAvg** (McMahan et al., 2017): The traditional centralized federated learning method lacks computational efficiency, communication efficiency, and adaptability to data heterogeneity.
- **Ditto** (Li et al., 2021b): A centralized personalized federated learning framework designed to handle data heterogeneity by training both global and local models with regularization techniques. This approach enhances robustness and adaptability to the non-i.i.d. challenge but does not improve computational or communication efficiency.
- **FedDST** (Bibikar et al., 2022): A communication and computation-efficient federated learning framework managed by a central server. Clients share models with the same layer-wise sparsity distribution and perform dynamic pruning on the central server. More details are clarified in Section 5.5.

- **GossipFL** (Tang et al., 2022): A decentralized gossip learning framework with a central coordinator that shares the seed for pruning masks collaboratively. It uses a mixing or adjacency matrix based on communication information such as bandwidth to achieve efficient peer-to-peer communication without improving computational efficiency or adaptability to data heterogeneity.

- **DFedAvgM** (Sun et al., 2022): A decentralized version of FedAvg with momentum, using the Adam optimizer with quantization to achieve communication efficiency but lacking computational efficiency and adaptability to data heterogeneity.

- **DisPFL** (Dai et al., 2022): A reduced version of DA-DPFL, without new scheduling and further pruning methods, serving as a significant competitor to the proposed method. It achieves personalization with communication and computation efficiency through mask-based aggregation on the aggregator. The pruning strategy is similar to FedDST.

- **BEER** (Zhao et al., 2022): A decentralized stochastic learning framework with gradient tracking and compression, which accelerates convergence but does not improve computational efficiency or adaptability to data heterogeneity.

- **DFedSAM** (Shi et al., 2023): A decentralized federated learning method with the SAM (Sharpness-Aware Minimization) optimizer, searching for models with uniformly low loss values to accelerate aggregation and handle data heterogeneity without enhancing computational or communication efficiency.

6.5.3 Configurations

System Configuration

Experiments are performed on a system with $M = 100$ clients, each possessing sufficient computational power to train the specified models locally, representing more capable edge devices such as IoT sensors or smartphones. These clients are assumed to have intermittent yet reliable network connectivity, as determined by the chosen topology for model updates, while securely storing data locally to ensure privacy. Following the methodologies in (Shi et al., 2023; Dai et al., 2022), DFL is configured to mirror the highest communication burden on the central server. Specifically, the number of connected clients in CFL (neighbor size $C = 10$) is set to 10, and in DFL, each client acts as a central server, connecting with $C = 10$ other clients. Unlike CFL, all DFL baselines except DFedSAM are configured to have half the communication cost. The sparse training methods, including DA-DPFL, FedDST, and DisPFL, are initialized with a sparsity $s_i^0 = 0.5$ for all clients $i \in [M]$. The main experiments are conducted using a *randomly time-varying* connected topology

(Figure 6.3.e). Additional experiments are conducted using *ring* (Figure 6.3.b) and *fully-connected* (Figure 6.3.c) topologies to verify the robustness of the proposed method. The total number of communication rounds varies across experiments: $T = 300$ for HAM10000 and $T = 500$ for the other two datasets.

Training Details

To ensure fair comparisons, experimental hyperparameters are aligned with the setups in (Dai et al., 2022; Shi et al., 2023). Unless otherwise specified, the number of local epochs is fixed at 5 for all approaches, and a Stochastic Gradient Descent (SGD) optimizer with a weight decay of 5×10^{-4} is used. The learning rate is initialized at 0.1 and decays exponentially by a factor of 0.998 after each global communication round. The batch size is consistently set to 128 for all experiments. Global communication rounds are set to 500 for CIFAR10 and CIFAR100, and 300 for HAM10000. Parameters are set as $\delta_v = 0.5, b = 0, c = 1.3$, and $\{p, q, \gamma, \eta_c\} = \{0.5, 1, 0.9, 1\}$ following Diao et al. (2022), with δ_{pr} values in $\{0.01, 0.02, 0.03\}$ for all experiments.

In the CFL baseline implementation, Ditto's local training is divided into two phases: a global model training phase for 3 epochs and a personalized model training phase for 2 epochs. Additionally, the update mask reconfiguration interval in FedDST is determined by a grid search over $[1, 5, 10, 20]$. In the DFL setup, when compression techniques are incorporated, the busiest communication load is halved. GossipFL utilizes a *Random Match* approach, randomly clustering clients into groups. Momentum SGD is used in DFedAvgM and DFedSAM, with a momentum factor $\beta = 0.9$. A grid search determines the ρ value for DFedSAM from $[0.01, 0.02, 0.05, 0.1, 0.2, 0.5]$, based on the SAM optimizer configuration.

Energy Cost Simulation

For cost analysis, an NVIDIA 4090 GPU with 80 TFLOPS and 450 W TDP serves as the standard to evaluate clients' computational power and energy consumption. The system configuration assumes a bandwidth of 1 Gbps, with each client network card consuming 1 W, as referenced from (Feeney and Nilsson, 2001). The metric C_{time} is extracted from Table 6.4, and both C_{energy} and C_{time} are converted to monetary units using the formulas $(1-\theta)\$/s$ and $\theta\$/J$. To reconcile the discrepancy between theoretical and actual GPU execution times, real-world algorithm executions are conducted on the GPU. These executions reveal that actual times are five times longer than theoretical predictions, necessitating a correction factor of 5 for computation time, calculated as $T_{\text{comp}} = 5 \times \frac{D_{\text{FLOP}}}{V_{\text{FLOPS}}}$, and for energy, $C_{\text{comp}} = T_{\text{comp}} \times P_{\text{comp}}$. For communication time estimation, the formula $T_{\text{comm}} = \frac{D_{\text{comm}}}{B}$ is used, where D_{comm} represents the data volume to be transferred and B indicates the system's total bandwidth. Consequently, the communication energy cost

C_{comm} is calculated as $C_{\text{comm}} = T_{\text{comm}} \times P_{\text{comm}}$, with P_{comm} representing the transmission power of the wireless network card.

6.6 Experimental Analysis

This section presents a comprehensive analysis of the experimental results, focusing on cost efficiency and learning efficiency.

6.6.1 Cost Efficiency

The efficiency of the proposed algorithm is assessed by examining two primary metrics: the Floating Point Operations (FLOP) needed for inference and the communication overhead during convergence rounds. To ensure a consistent baseline, the initialization protocol from DisPFL is used, which standardizes the initial communication costs and FLOP values during the early pruning phase of training.

The pruning stages in DA-DPFL significantly reduce these costs. The final sparsity levels achieved are $(0.61, 0.56)$ for HAM10000, $(0.65, 0.73)$ for CIFAR10, and $(0.70, 0.73)$ for CIFAR100 under Dir. and Pat. partitioning, respectively. These results are achieved within the specified communication rounds. Notably, the busiest communication costs and training FLOPs for DA-DPFL are lower than those of the most efficient DFL baseline, DisPFL.

Table 6.4 on busiest communication cost and final training FLOPs highlights the efficiency of DA-DPFL. For HAM10000, CIFAR10, and CIFAR100, DA-DPFL achieves significantly lower communication costs and FLOPs compared to other methods, indicating its superior efficiency in both communication and computation.

To evaluate potential delays in DA-DPFL, the total cost C_{total} is computed as defined in Luo et al. (2021) and Zhou et al. (2022):

$$C_{\text{total}} = (1 - \theta)C_{\text{time}} + \theta C_{\text{energy}},$$

where $\theta \in [0, 1]$ is set to 0 for highly time-sensitive applications and to 1 for energy-sensitive tasks; this metric unifies time and energy costs into monetary units (USD \$).

To provide a realistic perspective on DA-DPFL's introduction, communication and computation costs (FLOP) are combined into energy expenditure:

$$C_{\text{energy}} = C_{\text{comm}} + C_{\text{comp}},$$

where C_{comm} and C_{comp} represent communication and computational costs, respectively. Figures 6.6 and 6.7 depict the cost-effectiveness of DA-DPFL compared to other DFL

baselines. When $\theta \to 0$ initially, DA-DPFL incurs higher time costs. However, as θ increases beyond 0.2, DA-DPFL shows substantial advantages over other DFL algorithms (represented by solid lines), with its benefits growing as θ increases further. CFLs, due to system configuration, exhibit significantly lower communication (1%) and computation (10%) costs compared to DFL, which overall shows better cost efficiency but slower convergence. Even considering waiting time, DA-DPFL achieves both cost efficiency and learning efficiency (convergence speed).

Table 6.4: Busiest Communication Cost & Final Training FLOPs of All Methods

	HAM10000		CIFAR10		CIFAR100	
Algorithm	**Com. (MB)**	**FLOP (1e12)**	**Com. (MB)**	**FLOP (1e12)**	**Com. (MB)**	**FLOP (1e12)**
FedAvg	887.8	3.6	426.3	8.3	353.3	2.3
Ditto	887.8	3.6	426.3	8.3	353.3	2.3
FedDST	443.8	2.0	223.1	7.1	176.7	1.6
GossipFL	443.8	3.6	223.1	8.3	176.7	2.3
DFedAvgM	443.8	3.6	223.1	8.3	176.7	2.3
DisPFL	443.8	2.0	223.1	7.1	176.7	1.6
BEER	443.8	3.6	223.1	8.3	176.7	2.3
DFedSAM	887.8	7.2	426.3	17.0	353.3	4.6
DA-DPFL_Dir	**346.2**	**1.9**	**149.1**	**4.1**	**107.7**	**1.0**
DA-DPFL_Pat	**394.4**	**2.0**	**115.1**	**3.8**	**94.8**	**0.9**

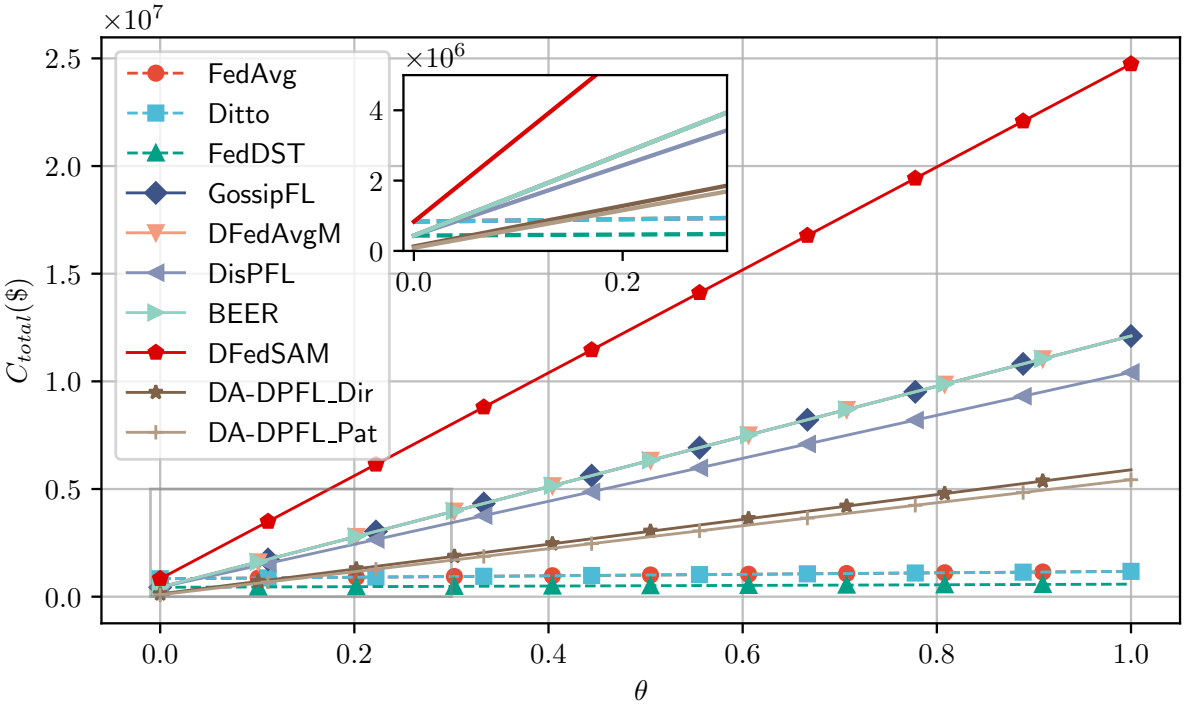


Figure 6.6: Total cost (energy and time cost, in USD) of DA-DPFL and all baselines evaluated on CIFAR10 against θ.

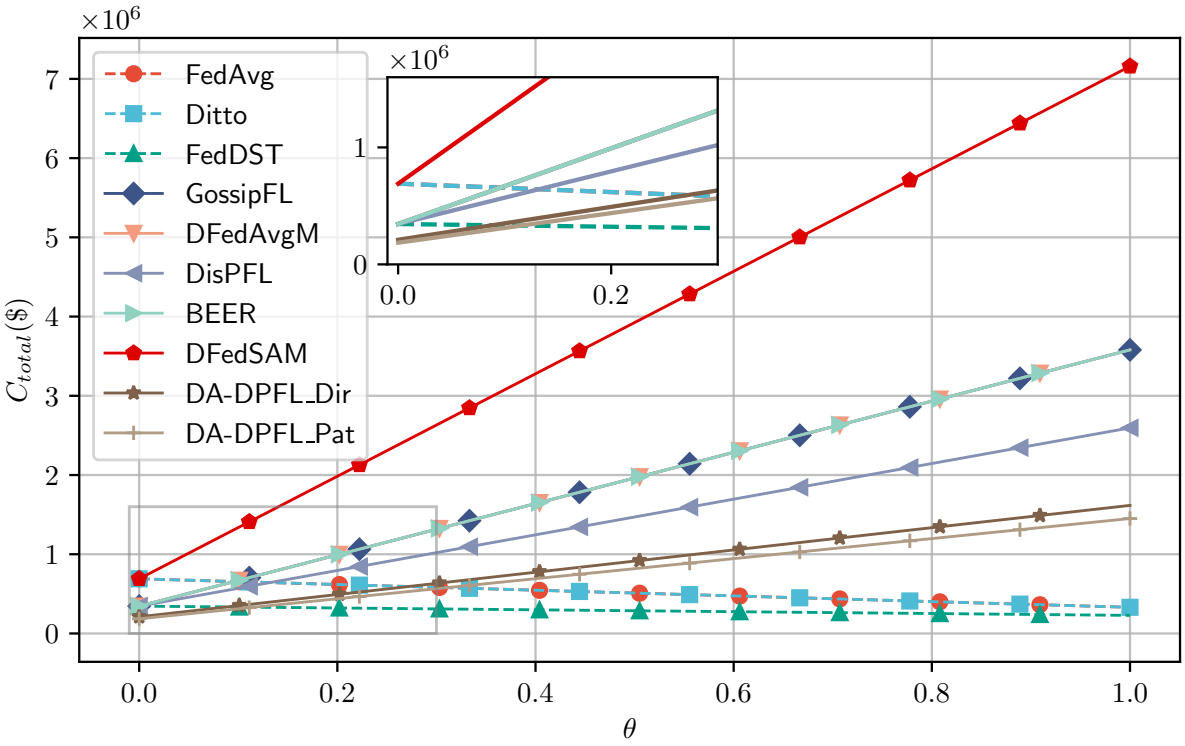


Figure 6.7: Total cost (energy and time cost, in USD) of DA-DPFL and all baselines evaluated on CIFAR100 against θ.

6.6.2 Learning Efficiency

DA-DPFL surpasses all baselines in *top*-1 accuracy in five out of six scenarios, demonstrating robustness under extreme non-iid conditions ($n_{cls} = 2$) (Fig. 6.9, Table 6.5). It outperforms the next best DFL baselines (DisPFL and GossipFL) by $2-3\%$, with a slight shortfall in HAM10000 ($n_{cls} = 2$) by 0.5% compared to DFedAvgM. DA-DPFL consistently exceeds DisPFL in sparse model training and generalization while maintaining efficient convergence. In contrast, CFL lags in convergence due to limited client participation per round. Momentum-based methods like DFedAvgM accelerate initial learning, while BEER, with gradient tracking, shows rapid convergence but does not necessarily reduce generalization error. DA-DPFL balances convergence rate and generalization performance, outperforming other baselines and achieving target accuracy with reduced costs.

Additional experiments using *ring* and *fully-connected* (FC) topologies provide further evidence of DA-DPFL's adaptability. As shown in Table 6.6, DA-DPFL achieves the highest accuracy and sparsity levels compared to other methods in both topologies. In the *ring* topology, DA-DPFL attains an accuracy of 69.83% with a sparsity of 0.65, significantly outperforming DisPFL (67.65%) and other baselines. Similarly, in the *fully-connected* topology, DA-DPFL reaches an accuracy of 89.11% with a sparsity of 0.68, surpassing DisPFL's 86.54% and other baselines. These results highlight DA-DPFL's superior per-

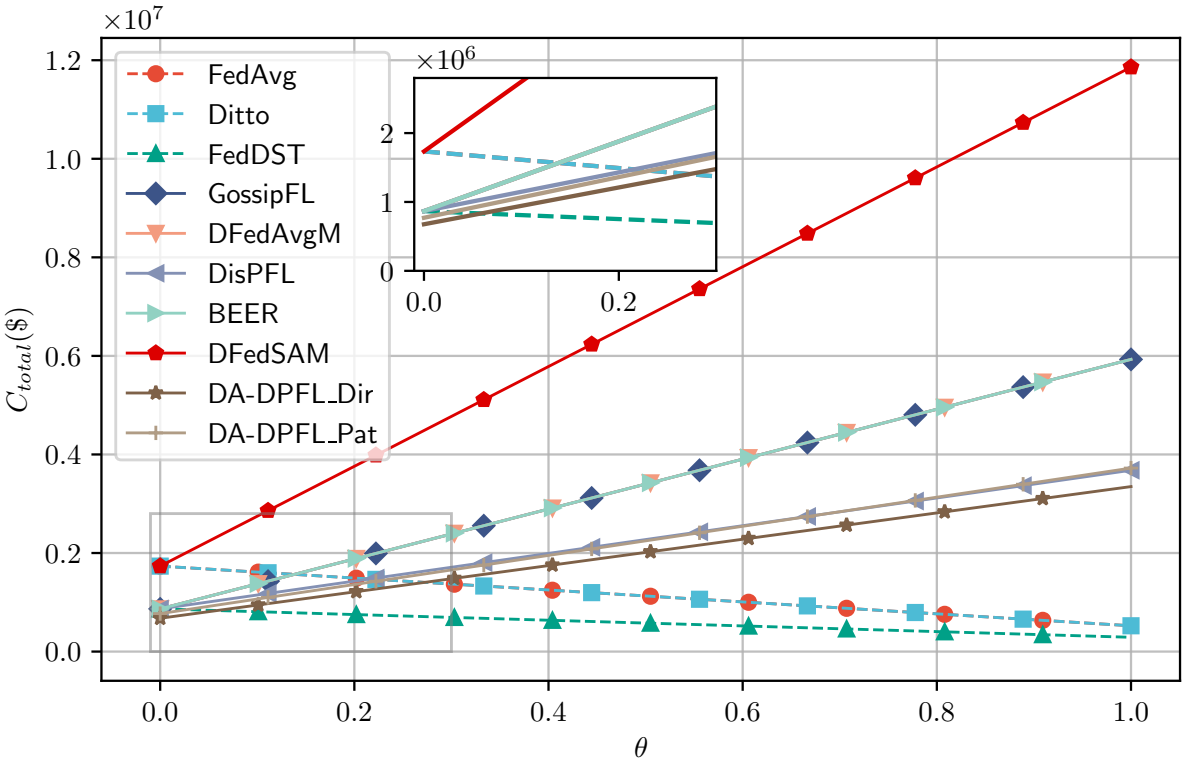


Figure 6.8: Total cost (energy and time cost, in USD) of DA-DPFL and all baselines evaluated on HAM10000 against θ.

formance and efficiency in different network configurations.

The accuracy comparison across different datasets (Table 6.5) further reinforces the superior performance of DA-DPFL. It achieves the highest accuracy in most cases, particularly excelling in CIFAR10 and CIFAR100 datasets under both Dir. and Pat. partitioning. This demonstrates the robustness and effectiveness of DA-DPFL in handling diverse and challenging data distributions.

In summary, DA-DPFL maintains a significant lead in performance across various scenarios, demonstrating both high accuracy and efficient model sparsity within 500 communication rounds.

6.6.3 Hyperparameter Analysis

Analysis on Neighborhood Size C

The impacts of the hyper-parameter C on the training efficiency of DA-DPFL are investigated by training ResNet18 on CIFAR10. The neighborhood size parameter C significantly affects the scheduling efficiency, where a higher C value enhances convergence by increasing the reuse of trained models throughout the training process, although it introduces the risk of potential delays. As depicted in Fig.6.10 (Bottom), $C = 20$ slightly surpasses

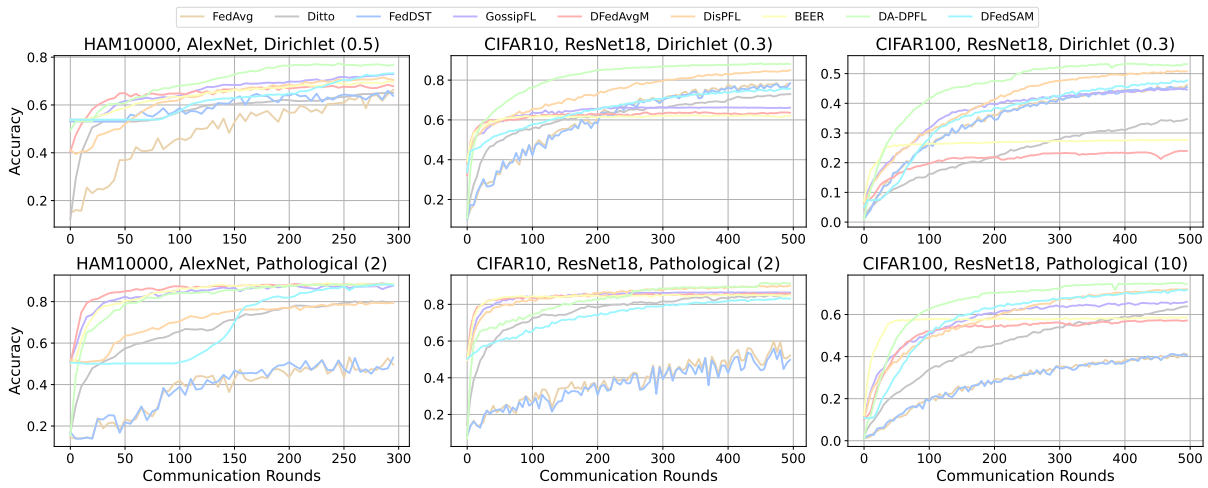


Figure 6.9: Test (*top-1*) accuracy of all baselines, including CFLs and DFLs, across various model architectures and datasets.

Table 6.5: Accuracy Comparison Across Different Datasets

Method	**HAM10000**		**CIFAR10**		**CIFAR100**	
	Dir. (0.5)	**Pat. (2)**	**Dir. (0.3)**	**Pat. (2)**	**Dir. (0.3)**	**Pat. (10)**
FedAvg	65.92 ± 0.3	55.68 ± 0.4	79.30 ± 0.2	60.09 ± 0.2	46.21 ± 0.4	41.26 ± 0.3
Ditto	65.19 ± 0.2	80.17 ± 0.1	73.21 ± 0.2	85.78 ± 0.1	34.83 ± 0.2	64.41 ± 0.3
FedDST	66.11 ± 0.3	55.07 ± 0.4	78.47 ± 0.2	56.32 ± 0.3	46.01 ± 0.2	41.42 ± 0.2
GossipFL	72.92 ± 0.1	88.05 ± 0.1	66.43 ± 0.1	86.60 ± 0.1	45.09 ± 0.1	66.03 ± 0.1
DFedAvgM	68.30 ± 0.1	**88.89** ± 0.1	65.05 ± 0.1	85.34 ± 0.2	24.11 ± 0.1	57.41 ± 0.1
DisPFL	71.56 ± 0.1	80.09 ± 0.1	85.85 ± 0.2	90.45 ± 0.2	51.05 ± 0.3	72.22 ± 0.2
BEER	69.80 ± 0.1	88.75 ± 0.2	62.94 ± 0.1	85.48 ± 0.1	27.79 ± 0.1	58.71 ± 0.1
DFedSAM	73.74 ± 0.2	88.47 ± 0.3	75.74 ± 0.2	83.51 ± 0.1	47.86 ± 0.2	71.76 ± 0.1
DA-DPFL (*Ours*)	**76.32** ± 0.3	88.36 ± 0.3	**89.08** ± 0.3	**91.87** ± 0.1	**53.53** ± 0.2	**74.91** ± 0.1

$C = 10$ in performance, but it incurs nearly double the time delays.

Threshold δ_{pr}

Extending the total communication rounds from 500 to 1000, it is found that a target sparsity of $s = 0.8$ is achievable without sacrificing accuracy, with DA-DPFL achieving 89% accuracy compared to DisPFL's 83.27%. This result challenges the generalization gap assumption, indicating that precise initial sparsity ratio selection in fixed sparsity pruning is less critical as DA-DPFL achieves comparable or lower generalization error at higher sparsity levels through further pruning. Fig.6.10(Top) shows pruning decisions based on average detection scores across clients and their sparsity trajectories. The initially high detection score indicates a significant disparity between the random mask and the RigL algorithm-derived mask, contrasting with EarlyCrop's centralized, densely initialized model approach. After t^*, client models in DA-DPFL undergo incremental pruning, with the scale of pruning decreasing due to reduced model compressibility, as shown by changes

Table 6.6: DFL Performance Comparison for Ring and Fully Connected Topologies

Topology	Method	Acc (%)	Sparsity (s)
Ring	GossipFL	66.12 ± 0.1	0.00
	DFedAvgM	65.89 ± 0.1	0.00
	DisPFL	67.65 ± 0.2	0.50
	BEER	62.92 ± 0.1	0.00
	DFedSAM	66.61 ± 0.2	0.00
	DA-DPFL	**69.83** ± 0.3	0.65
FC	GossipFL	71.22 ± 0.2	0.00
	DFedAvgM	69.89 ± 0.1	0.00
	DisPFL	86.54 ± 0.2	0.50
	BEER	68.77 ± 0.1	0.00
	DFedSAM	79.63 ± 0.3	0.00
	DA-DPFL	**89.11** ± 0.2	0.68

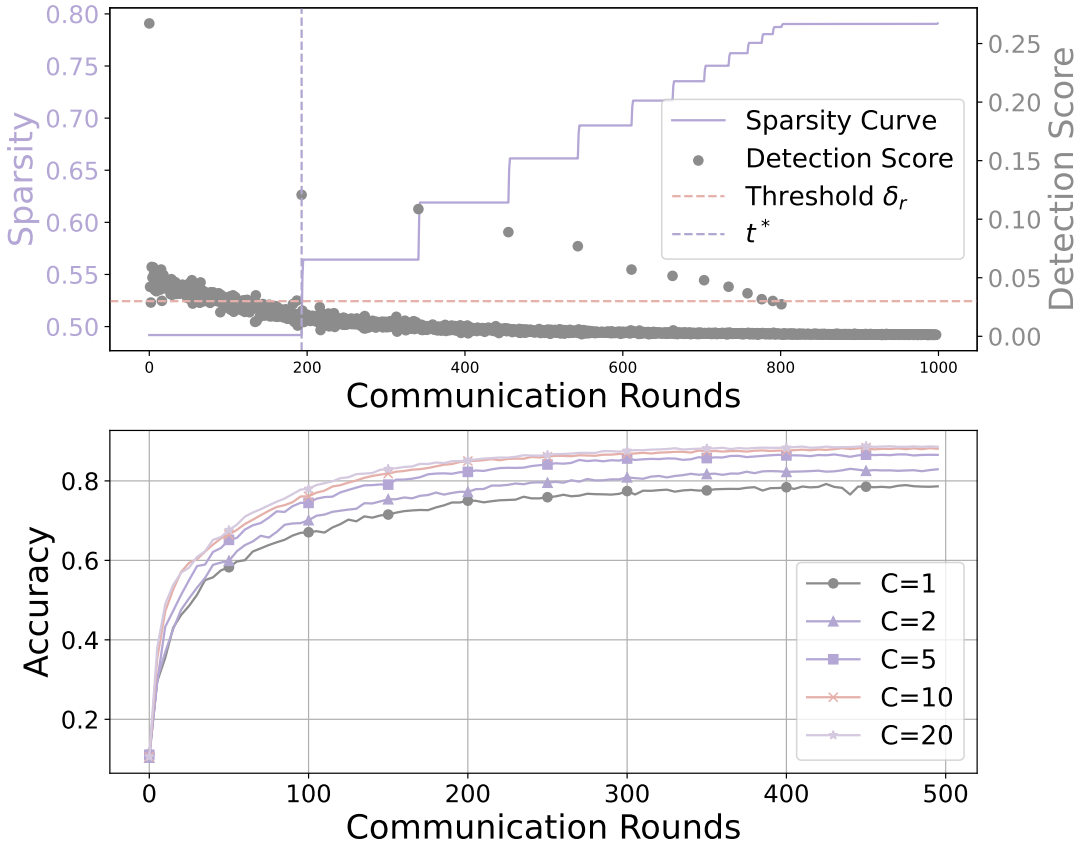


Figure 6.10: **(Top)** Relationship between sparsity and detection score; **(Bottom)** Impact of C involved in each training round on accuracy (CIFAR10, $Dir(0.3)$, $\delta_{pr} = 0.03$).

in sparsity at each pruning phase.

To examine the effect of the early pruning threshold δ_{pr}, experiments were conducted using CIFAR10 and ResNet18. Fig.6.11 highlights the importance of pruning timing,

revealing optimal thresholds vary for different data partitions and their corresponding detection score differences. Early pruning, while accelerating sparsity achievement, hampers crucial learning phases, whereas excessively delayed pruning is akin to post-training pruning, which incurs higher costs. Results suggest early-stage further pruning between 30-40% of total communication rounds, with a threshold range of $0.02 - 0.03$, to balance model performance and energy efficiency.

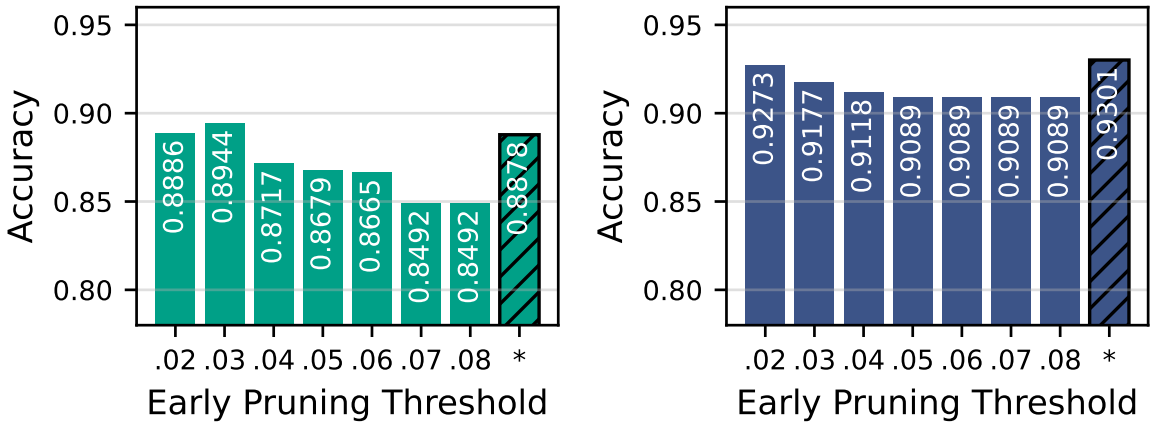


Figure 6.11: Impact of δ_{pr} on final prediction accuracy of achieving sparsity $s = 0.8$ with CIFAR10 ($C = 10$) *Dir* (left) and *Pat* (right) partitions (where * indicates DA-DPFL without further pruning, i.e., fixed sparsity $s = 0.5$).

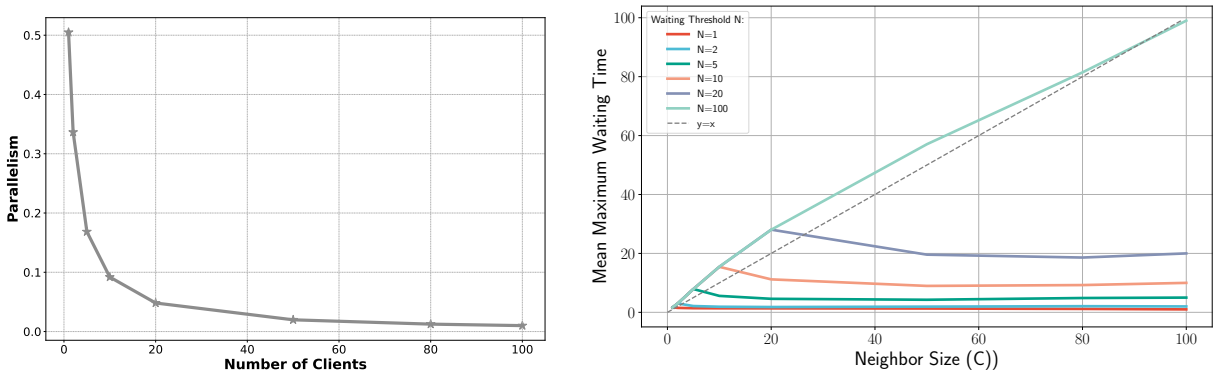
Parallelism and Delay Analysis and Threshold N

This section presents the impact of C on parallel computing without a constraint N and then analyzes the effect of N subsequently.

To estimate the average impact on parallelism and latency due to waiting times, 10,000 iterations were conducted. Parallelism is defined as the proportion of clients starting training concurrently. Figure 6.12a shows a decline in parallelism as the neighborhood size C increases (with $K = 100$ clients). Figure 6.12b presents delay against C. The black line, representing $N = C$, delineates the outcome of waiting for the most delayed clients, scaling almost linearly with the neighborhood size C. Figure 6.12b also demonstrates the effectiveness of the constraint $N \leq C$ in reducing delays while increasing C.

In cases where $C = N = 100$, DA-DPFL transitions to sequential learning, while with $C = 100$ and $N = 2$, DA-DPFL maintains high parallelism with opportunities for model reuse. With $N = 0$, DA-DPFL reverts to DisPFL with our pruning strategy.

For a fair comparison, different waiting thresholds $N = \{0, 2, 5\}$ are evaluated with the same experimental setup as in 6.5.3 for the *Dirichlet* partition. Results in Fig. 6.13 demonstrate that increasing N consistently enhances model accuracy across CIFAR10 and CIFAR100 datasets, with CIFAR10 showing up to a 1.87% increase and CIFAR100 a 1.41%



(a) (b)

Figure 6.12: (a)Impact of neighborhood size C on parallelism and delay; (b)Characteristic of proposed time-varying connected topology: delay caused by waiting across neighbor size C and waiting control threshold N.

increase in accuracy from $N = 0$ to $N = 10$. Interestingly, the HAM10000 dataset shows no significant improvement beyond $N = 5$, suggesting that task-specific characteristics influence the optimal N selection. Even the performance for $N = 0$ cases surpasses DisPFL, demonstrating the effectiveness of our further pruning strategy. Additionally, model reuse redundancy can occur, especially with large C. Selecting N allows for a trade-off between waiting time and model performance.

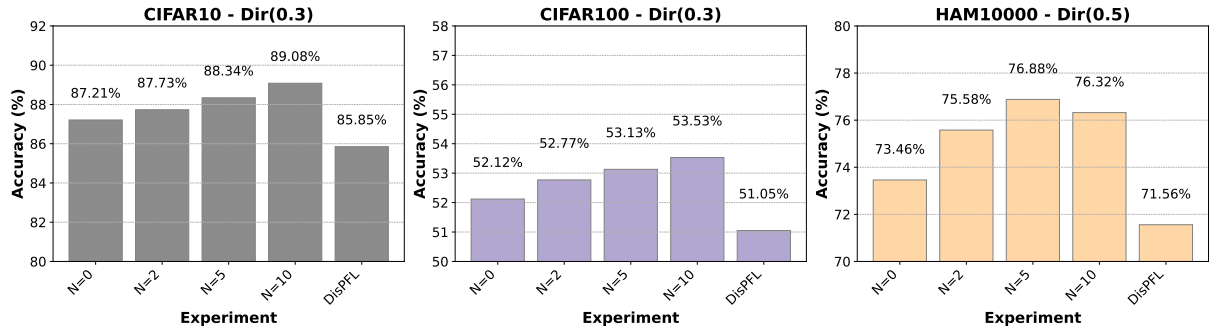


Figure 6.13: Performance with different waiting threshold N

6.7 Limitations and Future Research

While DA-DPFL presents a significant advancement in decentralized federated learning, it is not without limitations. Below, we discuss these limitations and corresponding future research directions to address them.

Most federated learning (FL) algorithms, including DA-DPFL, primarily focus on time-invariant data distributions, limiting their applicability in environments where data distributions evolve over time. Extending DA-DPFL to handle non-stationary data distributions is crucial. Developing algorithms that can dynamically adjust model parameters

and pruning levels in response to changing data distributions is essential. Techniques from continual learning and meta-learning could be particularly useful in this context.

Additionally, the assumption of uniform client availability and stable communication links is often unrealistic. Although the decentralized federated learning framework of DA-DPFL is robust, the contributions of dynamic pruning and aggregation might be influenced by client variability and unstable communication links. Clients may frequently join and leave the network, and communication links may be unstable, adversely affecting DA-DPFL's performance. Therefore, developing adaptive mechanisms to handle client variability and unstable communication links is imperative. Alongside the reuse index, incorporating dynamic client selection strategies based on network conditions can help tolerate communication disruptions and client dropouts, ensuring the framework's robustness and effectiveness.

6.8 Conclusions

This chapter extends the utilization of pruning to achieve efficiency in centralized federated learning, which is in Chapter 5, to optimize the efficiency in decentralized federated learning, i.e., a more flexible and robust framework. The DA-DPFL algorithm comprises two main components. The innovative dynamic aggregation strategy facilitates the reuse of knowledge from other clients within the same training round, creating a hybrid (parallel and sequential) learning paradigm. This approach accelerates training by reducing the number of rounds needed to achieve target accuracy. Additionally, the further pruning method, orthogonal to the fixed pruning strategy, reduces training costs related to communication and computation. It balances pruned model performance and energy efficiency by automatically determining the optimal pruning time.

As demonstrated in Section 6.6, and compared against competitive baselines ranging from centralized to decentralized federated learning across various tasks (data, model, and topology), DA-DPFL achieves energy efficiency while enhancing model generalization performance. It does this by finding effective masks to achieve personalization, thereby addressing data heterogeneity challenges. Furthermore, the proposed learning paradigm is supported by a solid theoretical analysis of convergence.

Thus, it can be concluded that efficiency and effectiveness can be enhanced through pruning in distributed machine learning. The successful application of pruning to improve efficiency in regular-sized neural networks motivates us to explore its potential for foundation models, i.e., larger-scale models. The next Chapter 7 investigates the benefits and challenges of integrating federated learning with diffusion models and other statistical learning techniques as future work and concludes the whole thesis.

Chapter 7

Conclusions & Future Research

7.1 Conclusions

This thesis explores the development and optimization of collaborative distributed machine learning frameworks, with a particular focus on knowledge reuse and model sparsification in distributed computing paradigms, including edge computing and federated learning. By addressing critical challenges such as computational overhead, communication costs, data heterogeneity, and privacy concerns, this work significantly advances the state of the art in distributed machine learning.

The contributions of this research extend beyond purely technical domains. The proposed frameworks and methodologies are critical in enabling intelligent systems for energy-sensitive, privacy-preserving, and resource-constrained environments, such as IoT networks, healthcare systems, and smart city applications. By improving the efficiency of distributed systems and reducing resource consumption, this work supports the deployment of sustainable and scalable AI solutions in diverse real-world scenarios. These advancements empower broader access to AI capabilities, particularly in areas with limited computational resources, fostering more equitable technological growth.

Through these contributions, the research bridges technical innovation and societal impact, paving the way for more efficient, accessible, and responsible distributed machine learning frameworks. We summarize the contributions of this work in detail:

7.1.1 Summary of Contributions

- **Knowledge Reuse in Edge Computing:** We introduce innovative methods for reusing pre-trained models in edge computing environments, specifically the Borrower-Loaner-Matching (BLM) mechanism and the Model-Reusability-Monitoring (MRM) system.
  - Due to privacy concerns, sending data from the data owner to other parties is

prohibitive. The feasibility of reusing models relies on the assumption of task similarity, i.e., data similarity. To measure this similarity privately, we propose using statistical synopses, including Maximum Mean Discrepancy (MMD) and Cosine Dissimilarity on the largest eigenvector. This approach transforms the building of BLM into two statistical hypothesis tests in edge computing environments.

- Data is not always static; therefore, we also propose the MRM system to ensure the effectiveness of source models, which are labeled as still reusable for target tasks under data streams. We empirically find that the Sum of Squares Error (SSE) is highly correlated with MMD and theoretically show that the *online* computation of SSE is more lightweight than MMD. Additionally, we transform the MRM model into a statistical hypothesis test with statistics calculated for Holt-Winter Models.
- Extensive results from various scenarios validate the predictions of BLM, which avoids redundant computation in edge computing and ensures scalability since the reusable model achieves accuracy comparable to independently trained models locally. Concurrently, the MRM system correctly identifies concept drift, violating the source model's reusability assumption. Hence, maintenance and efficiency in distributed computing systems can be achieved through the proposed approaches.

- **Enhanced Reusability in Distributed Multi-task Learning (DMtL):** We extended the concept of reuse to multi-task learning scenarios, which introduces a two-phase framework that clusters tasks based on performance metrics derived from Partial Learning Curves (PLC) with Gaussian Mixture Models (GMM) and assigns tasks to representative head of groups to conduct DMtL. The assumption of the existence of a pre-trained model might not always hold. This motivates us to derive an efficient method for distributed machine learning with the concept of knowledge reuse. Then, considering a distributed system with several M clients, the objective is to train only $K \ll M$ reusable models efficiently, and such reusable models have comparable or better predictability. Then the questions raise that (1) firstly *how can we decide K models are reusable for M clients* and (2) subsequently *how can we enable the K trained models to perform better than their local trained models?*
  - In the clustering phase, inspired by meta-learning, we select Partial Learning Curves (PLC) as meta-features to determine the *similarity* among tasks of different clients. This approach aids in selecting the most *representative* client for subsequent training and answering the first question above. The following assumptions support our methodology (as stated in Chapter 4.3.1): (1) **Data**

Sufficiency: We assume that the data from a single client is sufficient for training the model. This implies that a model trained on the data from one client will exhibit performance comparable to a model trained on the aggregated data from the entire group of clients, i.e., $\mathbb{E}_{D_i}[f(x)] \approx \mathbb{E}_{D_{group}}[f(x)]$ for a model f, indicating that the expected performance of the model trained on client i's data is comparable to that trained on the group's data; (2) **Data Distribution Similarity**: We assume that the data distribution of the selected client is representative of the entire group. This ensures that the model trained on this client's data is reusable across the group, maintaining performance consistency, i.e., $D_i \approx D_{group}$, indicating that the statistical properties of the data held by the representative client are similar to those of the entire group, thus ensuring model reusability. These assumptions facilitate the effective selection of a representative client, thereby enhancing the efficiency and scalability of the distributed training process.

- In the dissemination phase, driven by multitask learning, we develop the Distributed Multi-task Learning (DMtL) framework to enhance model reusability by improving model performance through joint learning from distinct data distributions. This addresses the second problem, as stated above. Note that, after clustering, each client holds an intermediate model where the training epochs/rounds are only conducted for a *partial* segment of the entire process. Based on the clustering results and the selection of representative heads, models can be safely trained collaboratively among K heads rather than N clients. Hence, this approach achieves efficiency by (1) reducing the number of clients required in distributed multi-task learning and (2) avoiding redundant training of models under our assumptions.
- We simulate scenarios under our assumptions and conduct experimental evaluations on real datasets. Specifically, the proposed two-phase learning framework outperforms both single-task learning and other multi-task learning approaches regarding model accuracy and the newly defined reusability metrics within and outside clusters. As discussed above, computation and communication efficiency are achieved by reducing the number of required clients during training. Overall, extensive results showed that the proposed two-phase DMtL framework improves model performance in classification tasks, reduces training costs, and accelerates the whole training process in scale, even under non-i.i.d. data partition settings.

- **Pruning in Federated Learning:** To address the challenges of high computational demands for resource-constrained devices, we integrate pruning techniques in federated learning to obtain extremely sparse models. This approach reduces model

sizes by eliminating redundant parameters and decreasing computational and communication costs without compromising model performance. The proposed FedDIP integrates dynamic pruning with error feedback and incremental regularization in federated learning. Traditional one-shot pruning to achieve high sparsity before training often results in significant model degradation despite offering computation and communication efficiency. Conversely, post-training pruning lacks these efficiency advantages. Hence, a method is required to balance model performance with communication and computation efficiency, termed dynamic pruning. Existing dynamic pruning techniques either maintain a fixed sparsity with consistent mask updates to find the *lottery ticket*, i.e., the optimal mask, or prune the dense model from initialization, sometimes necessitating initial training on certain devices. The first method shares the drawbacks of one-shot pruning, particularly at high sparsity levels, while the second method's reliance on dense model initialization can be prohibitive in some cases.

- To address information loss in the first method, we propose dynamic pruning with error feedback. This technique allows pruned connections to be recovered by adding back error terms, providing more information than methods with fixed sparsity, thus facilitating extreme pruning.
- Magnitude-based pruning methods guide which components, such as weights or neurons, are important. However, the diverse value ranges across different layers are problematic. Regularization-based pruning methods address this by shrinking the feature space, forcing unimportant values to approach zero. Wang et al. (2021) demonstrated that growing regularization exploits Hessian information for more accurate pruning without knowing their values. This accurate pruning from the growing penalty term is, therefore, beneficial for extreme pruning in federated learning. The integration of dynamic pruning with error feedback and growing regularization has the potential to find a mask with extreme sparsity in federated learning.
- The efficiency of FedDIP is verified through various experiments. Compared with state-of-the-art methods, FedDIP maintains the highest accuracy when the target sparsity exceeds 0.8. Accurate pruning enables the pruning of the first and last layers, often left unpruned by other methods, thus enhancing opportunities for extreme pruning. Within the same communication budget, FedDIP achieves comparable model performance. Consequently, FedDIP efficiently prunes neural networks through distributed training, maintaining model performance while reducing computation, communication, and storage requirements.

- **Dynamic Aggregation and Pruning in Decentralized Federated Learning:** We expand the utilization of pruning in centralized federated learning (CFL) to decentralized federated learning (DFL). CFL and DFL share the same common challenges, which are the core issues in this thesis, including expensive communication and computation costs and statistical heterogeneity. Compared with CFL, DFL offers several advantages such as high maintenance and robustness against adversarial attack due to the avoidance of the orchestration of a central server; various topologies of DFL increase the flexibility to be deployed on heterogeneous system architectures; faster convergence speed because of all clients participating in training (DFedAvgM, Sun et al. (2022)) unlike the CFL that select a subset of clients participate in training. However, such increased client participation results in higher total communication and computation costs. Does the question raise that *can the challenges of decentralized, federated learning be mitigated while retaining its advantages?* To address such a question, we propose a DA-DPFL algorithm, which leverages the dynamic aggregation policy and an innovative, dynamic sparse-to-sparser training scheme conditioned on the model compressibility.

  - The data heterogeneity is solved with personalization with personalized masks, which is another advantage of pruning.
  - Since all clients participate in training, the redundancy in computation for clients exists. Sequential federated learning (SFL) accelerates the training by transferring sequential knowledge from one to another. However, the fixed order in SFL induces the problem considering data heterogeneity (Yuan et al., 2024). We propose to leverage the concept of knowledge reuse to create a hybrid scheme of sequential and parallel federated learning for model aggregation, i.e., dynamic aggregation. The mechanism is controlled by the random reuse index and a waiting threshold to control the delay caused by the waiting, which often appears in SFL. The required round to achieve convergence is reduced because of reusing model updates from the previous clients in the same training round.
  - The question discussed in pruning in CFL also holds in DFL that the pruning strategy balances the trade-off between the model performance and efficiency. To relax the careful selection of initial sparsity for the fixed sparsity pruning method in DFL, we provide a sparse-to-sparser training method by (1) measuring the model compressibility and (2) computing the detection score for optimal pruning time.
  - The efficiency of DA-DPFL is validated through comprehensive experiments. Compared with existing work, DA-DPFL achieves superior accuracy across various datasets and topologies, demonstrating its robustness and scalability. The

dynamic adjustment of both aggregation and pruning processes ensures efficient utilization of computational resources and maintains high model performance. In experiments with ring and fully-connected topologies, DA-DPFL outperformed other methods in accuracy and achieved higher sparsity levels. The algorithm's ability to handle diverse data distributions further highlights its effectiveness. Within the same communication rounds, DA-DPFL delivers better model performance and less energy costs, making it suitable for large-scale distributed energy-sensitive applications. Therefore, DA-DPFL balances effectiveness and energy efficiency, reducing computational, communication, and storage costs.

7.2 Future Research Directions

In this section, we outline potential research directions to further enhance efficiency in distributed machine learning by leveraging knowledge reuse and sparsification.

7.2.1 Efficient Pruning Techniques for Foundation Models in Federated Learning

Foundation Models (FMs), such as Large Language Models (LLMs) and Diffusion Models (DMs), have demonstrated remarkable performance on complex tasks due to their large scale and capacity for capturing extensive knowledge. As stated in Zhuang et al. (2023), integrating FMs with Federated Learning (FL) offers a promising avenue to address challenges such as data scarcity, non-i.i.d. data distributions, and computational scalability. However, this integration presents several pressing challenges: (i) high communication and computational costs due to the size of FMs, (ii) inherent data heterogeneity in FL environments, and (iii) the need to adapt to continuously streaming data in a continual learning context.

Future research can address the outlined challenges by exploring innovative strategies aimed at enhancing efficiency and performance in federated learning frameworks. Adaptive sparsification protocols could play a pivotal role by dynamically identifying and removing less significant components, such as weights, neurons, or layers, during FL training. By leveraging sparsity-aware optimization techniques or reinforcement learning, these protocols can strike a balance between model accuracy and computational efficiency.

Additionally, hierarchical FL frameworks that utilize multi-tiered architectures—spanning clients, edge servers, and cloud infrastructure—offer a promising solution. For example, sparsified foundation models (FMs) could be trained locally on clients, aggregated at edge servers, and fine-tuned globally, thereby reducing communication costs while maintain-

ing robust model performance. Enhancing the quality of synthetic data generated by FMs through adaptive generative techniques, such as distribution-aware GANs or diffusion models, could further ensure alignment with client data distributions, mitigating the challenges posed by non-i.i.d. data.

Finally, integrating pruning methods with continual learning frameworks could enable FMs to efficiently adapt to streaming data. This approach minimizes memory and computational overhead while maintaining model relevance and performance in dynamic environments.

These directions aim to enable scalable and efficient FL training for FMs, addressing both system and data heterogeneity.

7.2.2 Optimizing Ensemble Learning through Pruning in Federated Learning

The proliferation of heterogeneous devices, such as mobile edges, edge servers, and cloud infrastructures, presents a need for distributed machine learning frameworks that optimize both latency and predictive performance. Current strategies face a trade-off: local models reduce latency but often lack predictive power, while cloud-based models achieve higher accuracy at the cost of increased latency.

Future work can explore innovative strategies to optimize ensemble learning within federated learning (FL), incorporating concepts such as transfer learning and knowledge reuse to further enhance efficiency and adaptability.

Pruned sub-model ensembles can be developed to create lightweight models tailored for ensemble learning. Pruning techniques could focus on the statistical contributions of individual sub-models to the ensemble's overall performance, eliminating redundant components while reducing computational and communication overhead. By leveraging transfer learning, pre-trained models can serve as a starting point for ensemble members, accelerating convergence and improving generalization across diverse tasks.

Task-aware ensemble optimization could further refine ensemble learning by integrating task-specific metrics into the pruning and aggregation processes. Models could be prioritized based on their performance on specific applications, such as low-latency tasks or high-accuracy scenarios, ensuring that the ensemble adapts effectively to the needs of diverse FL environments. Knowledge reuse mechanisms can also facilitate the transfer of learned representations between tasks, enabling efficient adaptation to new tasks or client data distributions.

Efficient aggregation algorithms that combine the strengths of traditional paradigms like bagging, boosting, and stacking could also be proposed. For example, dynamic stacking algorithms can adaptively select and combine pruned sub-models based on their performance across heterogeneous client data, enhancing robustness and predictive accuracy.

Finally, cross-tier ensemble models could be explored to leverage the unique capabilities of client, edge, and cloud levels. These models could distribute computational tasks intelligently across the tiers, optimizing latency, accuracy, and scalability. Incorporating transfer learning principles would allow knowledge gained at higher tiers (e.g., cloud-level models) to be effectively reused and adapted at lower tiers (e.g., client-level models), ensuring resource-efficient model development.

By addressing these directions, ensemble learning in FL can achieve greater resource efficiency, enhanced knowledge transfer, and high predictive performance across diverse applications and device capabilities.

Appendix A

Proofs

A.1 Proof for Chapter 3

The following is the proof for Theorem 1

Proof. Utilizing the prediction interval approach, which reflects the statistical likelihood of Z_t, we define the prediction interval for the HW model as follows:

$$\mathrm{Var}(e_t(\tau)) = \left[1 + \frac{1}{6}(\tau - 1)\xi_0^2(1 + \tau\xi_1 + \tau(2\tau - 1)\xi_1^2)\right] \mathrm{Var}(e_t(1)). \tag{A.1}$$

Given the threshold θ, the non-activation of BLM is upheld if:

$$\zeta_t + \tau b_t + 4.47\sqrt{\mathrm{Var}(e_t(\tau))} \leq \theta. \tag{A.2}$$

This inequality reflects the need to accommodate prediction intervals; thus, we employ 4.47 (Chebyshev's inequality) instead of the usual 1.96. The solution τ^* is found when:

$$y_t(\tau) = \theta - \zeta_t, \tag{A.3}$$

where $y_t(\tau) = \tau b_t + 4.47\sqrt{\mathrm{Var}(e_t(\tau))}$. The function $y_t(\tau)$ is approximated using the Newton-Raphson method to find τ^* efficiently. The decision rule is implemented by adjusting ξ_0, ξ_1 to minimize the squared forecasting error across the observed data: $\sum_{\tau=1}^{t}(Z_\tau - \hat{Z}_\tau)^2$. □

A.2 Proof for Chapter 4

A.2.1 Proof for Lemma 1

Proof. According to Smith et al. (2017) and Theorem 1 in Zhang and Yeung (2010), the objective $\mathcal{J}(\mathbf{W})$ in Equation (4.9) is jointly convex with respect to $\mathbf{W}$ and $\mathbf{\Omega}$, ensuring convergence via the SGD algorithm. □

A.2.2 Proof for Lemma 2

Proof. Consider a mini-batch SGD formula:

$$\min_{\boldsymbol{\omega}} \mathcal{H}(\boldsymbol{\omega}, \mathbf{Z}_i) := \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2 + \frac{1}{|\mathbf{Z}_i|}\sum_{j\in\mathbf{Z}_i} \mathcal{L}(\boldsymbol{\omega}, (\mathbf{x}_j, y_j)) \tag{A.4}$$

where $j \in \mathbf{Z}_i$ with $\mathcal{L}(\boldsymbol{\omega}, (\mathbf{x}_j, y_j)) = \max\{0, 1 - y_j\boldsymbol{\omega}^T\mathbf{x}_j\}$. The (sub-)gradient of weight $\boldsymbol{\omega}_i$ with respect to l is:

$$\Delta_i = \lambda\boldsymbol{\omega}_i - \mathbf{1}[y_j\boldsymbol{\omega}_i^T\mathbf{x}_j < 1]y_j\mathbf{x}_j, \tag{A.5}$$

where $\mathbf{1}[y_j\boldsymbol{\omega}_i^T\mathbf{x}_j < 1]$ is the indicator function. Hence, the update rule is:

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i - \eta_i\Delta_i. \tag{A.6}$$

At time i, we select the batch $\mathbf{Z}_i$ and set $\mathbf{Z}_i^* = \{j \in \mathbf{Z}_i : y_j\boldsymbol{\omega}_i^T\mathbf{x}_j < 1\}$. For the next epoch $i+1$, with learning rate η_i and substituting (A.5) into (A.4), we obtain:

$$\boldsymbol{\omega}_{i+1} = (1 - \eta_i\lambda)\boldsymbol{\omega}_i + \frac{\eta_i}{|\mathbf{Z}_i^*|}\sum_{j\in\mathbf{Z}_i^*} y_j\mathbf{x}_j, \tag{A.7}$$

which completes the proof. □

A.3 Proof for Chapter 5

A.3.1 Proof for Theorem 2

Before presenting the proof of Theorem 2, we emphasize the following relation:

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)}) - f(\bar{\boldsymbol{\omega}}'^{(t)})] = \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}^{(t+1)})] + \mathbb{E}[f(\bar{\boldsymbol{\omega}}^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})], \tag{A.8}$$

where the mask update occurs only at the server, and $\bar{\boldsymbol{\omega}}'^{(t)}$ is the global model received by the nodes at time t, marking the start of the local model training phase.

Lemma 3. *Given any mask function* $\mathbf{m} \in \{0,1\}^{n\times p}$ *for pruning, the Frobenius norm of the model weight/gradients matrix* $\boldsymbol{\omega}$ *is greater than or equal to that of the pruned one* $\mathbf{m} \odot \boldsymbol{\omega}$*, i.e.,*

$$\|\boldsymbol{\omega}\| \geq \|\mathbf{m} \odot \boldsymbol{\omega}\|. \tag{A.9}$$

Proof. According to the definition of the Frobenius norm, we have:

$$\|\mathbf{m} \odot \boldsymbol{\omega}\|^2 \;=\; \mathrm{Tr}([\mathbf{m} \odot \boldsymbol{\omega}]^T \cdot [\mathbf{m} \odot \boldsymbol{\omega}]) \;=\; \sum_{i=1}^{n}\sum_{j=1}^{p} |m_{ij}\boldsymbol{\omega}_{ij}|^2 \;\leq\; \sum_{i=1}^{n}\sum_{j=1}^{p} |\boldsymbol{\omega}_{ij}|^2 \;=\; \|\boldsymbol{\omega}\|^2.$$

□

Lemma 3 establishes that the quality of pruning δ_t is within the range $[0,1]$.

Lemma 4. *Given Definition 1 and Assumption 2, the effect of pruning on pruned model weights at the server* (δ_{t+1}) *is bounded as:*

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}^{t+1})] \leq \mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|]. \tag{A.10}$$

Proof. According to Assumption 2, we have:

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] \;-\; \mathbb{E}[f(\bar{\boldsymbol{\omega}}^{t+1})] \;\leq\; \mu\mathbb{E}[\|\bar{\boldsymbol{\omega}}'^{(t+1)} \;-\; \bar{\boldsymbol{\omega}}^{t+1}\|] \;=\; \mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|]. \tag{A.11}$$

□

Lemma 5. *Under the definitions provided in Section 5.4 and Assumptions 1 and 4,* $\mathbb{E}[f(\bar{\boldsymbol{\omega}}^{t+1})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})]$ *is bounded by:*

$$\begin{aligned}\mathbb{E}[f(\bar{\boldsymbol{\omega}}^{t+1})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})] \leq\; & \frac{(\gamma-1)L^2\eta_t^2 + \eta_t^2 L}{2C}\sum_{i=1}^{M}\rho_i\sigma_i^2 \\ & + \frac{(\gamma-1)\gamma E_l\eta_t^2L^2}{2}\sum_{k=t_c+1}^{t_c+E_l}\|\sum_{i=1}^{M}\rho_i\mathbf{v}_i'^{(k)}\|^2 - \frac{\eta_t}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 + \frac{\gamma\eta_t^2L-\eta_t}{2}\|\sum_{i=1}^{M}\rho_i\tilde{\mathbf{v}}_i'^{(t)}\|^2.\end{aligned} \tag{A.12}$$

Proof. Firstly, according to Assumption 1, we have:

$$\begin{aligned}\mathbb{E}[f(\bar{\boldsymbol{\omega}}^{t+1})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})] &\leq \mathbb{E}[\langle\bar{\boldsymbol{\omega}}^{t+1} - \bar{\boldsymbol{\omega}}'^{(t)}, \nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\rangle] + \frac{L}{2}\mathbb{E}\|\bar{\boldsymbol{\omega}}^{t+1} - \bar{\boldsymbol{\omega}}'^{(t)}\|^2 \\ &= \frac{\eta_t^2L}{2}\mathbb{E}\|\tilde{\mathbf{v}}'^{(t)}\|^2 - \mathbb{E}[\langle\eta_t\tilde{\mathbf{v}}'^{(t)}, \nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\rangle],\end{aligned} \tag{A.13}$$

where the equality holds because of (5.22).

Using the variance formula and the definition of $\bar{\mathbf{v}}'^{(t)}$, we can expand $\mathbb{E}\|\tilde{\mathbf{v}}'^{(t)}\|^2$ as follows:

$$\mathbb{E}\|\tilde{\mathbf{v}}'^{(t)}\|^2 \;=\; \mathbb{E}\|\tilde{\mathbf{v}}'^{(t)} - \mathbb{E}(\tilde{\mathbf{v}}'^{(t)})\|^2 + [\mathbb{E}(\tilde{\mathbf{v}}'^{(t)})]^2 \;=\; \mathbb{E}\|\tilde{\mathbf{v}}'^{(t)} - \bar{\mathbf{v}}'^{(t)}\|^2 + \|\bar{\mathbf{v}}'^{(t)}\|^2. \quad \text{(A.14)}$$

Considering the selection probability of clients P_t, we have:

$$\|\bar{\mathbf{v}}'^{(t)}\|^2 = \left\| \frac{1}{C}\mathbb{E}_{P_t}\left[\sum_{i\in P_t} \mathbf{v}_i'^{(t)}\right]\right\|^2 \leq \frac{1}{C}\mathbb{E}_{P_t}\left[\sum_{i\in P_t} \|\mathbf{v}_i'^{(t)}\|^2\right], \quad \text{(A.15)}$$

where the last inequality holds due to Jensen's Inequality.

Similarly, based on the definitions in (5.19), we have:

$$\begin{aligned}
\mathbb{E}[\|\tilde{\mathbf{v}}'^{(t)} - \bar{\mathbf{v}}'^{(t)}\|^2] &= \mathbb{E}\left[\left\|\mathbb{E}_{P_t}\left(\frac{1}{C}\sum_{i\in P_t}\tilde{\mathbf{v}}_i'^{(t)} - \frac{1}{C}\sum_{i\in P_t}\mathbf{v}_i'^{(t)}\right)\right\|^2\right] \\
&= \frac{1}{C^2}\left\{\mathbb{E}\left(\mathbb{E}_{P_t}\left[\sum_{i\in P_t}\|\tilde{\mathbf{v}}_i'^{(t)} - \mathbf{v}_i'^{(t)}\|^2\right]\right) + \sum_{i\neq j}\langle \tilde{\mathbf{v}}_i'^{(t)} - \mathbf{v}_i'^{(t)}, \tilde{\mathbf{v}}_j'^{(t)} - \mathbf{v}_j'^{(t)}\rangle\right\} \\
&= \frac{1}{C}\mathbb{E}\left[\sum_{i=1}^{M}\rho_i\|\tilde{\mathbf{v}}_i'^{(t)} - \mathbf{v}_i'^{(t)}\|^2\right]. \quad \text{(A.16)}
\end{aligned}$$

Substituting (A.15) and (A.16) into (A.14), and using Lemma 3, Definition 2, and Assumption 4, we get:

$$\begin{aligned}
\mathbb{E}\|\tilde{\mathbf{v}}'^{(t)}\|^2 &\leq \frac{1}{C}\mathbb{E}\left[\sum_{i\in P_t}\|\mathbf{v}_i'^{(t)}\|^2\right] + \frac{1}{C^2}\mathbb{E}\left[\sum_{i\in P_t}\|\tilde{\mathbf{v}}_i'^{(t)} - \mathbf{v}_i'^{(t)}\|^2\right] \\
&\leq \sum_{i=1}^{M}\rho_i\|\mathbf{v}_i'^{(t)}\|^2 + \frac{1}{C}\mathbb{E}\left[\sum_{i=1}^{M}\|\tilde{\mathbf{v}}_i'^{(t)} - \mathbf{v}_i'^{(t)}\|^2\right] \\
&\leq \sum_{i=1}^{M}\rho_i\|\mathbf{v}_i'^{(t)}\|^2 + \frac{1}{C}\sum_{i=1}^{M}\rho_i\sigma_i^2 \\
&\leq \gamma\sum_{i=1}^{M}\|\rho_i\mathbf{v}_i'^{(t)}\|^2 + \frac{1}{C}\sum_{i=1}^{M}\rho_i\sigma_i^2. \quad \text{(A.17)}
\end{aligned}$$

Next, we provide the boundary for $-\mathbb{E}[\langle \eta_t \tilde{\mathbf{v}}'^{(t)}, \nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\rangle]$:

$$\begin{aligned}
-\mathbb{E}[\langle \tilde{\mathbf{v}}'^{(t)}, \nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\rangle] &= -\langle \mathbb{E}[\frac{1}{C}\sum_{i\in P_t} \tilde{\mathbf{v}}_i'^{(t)}], \nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\rangle = -\langle \mathbb{E}[\sum_{i=1}^{M} \rho_i \tilde{\mathbf{v}}_i'^{(t)}], \nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\rangle \\
&\leq -\frac{1}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 - \frac{1}{2}\|\sum_{i=1}^{M} \rho_i \tilde{\mathbf{v}}_i'^{(t)}\|^2 + \frac{1}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)}) - \sum_{i=1}^{M} \rho_i \tilde{\mathbf{v}}_i'^{(t)}\|^2 \\
&= -\frac{1}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 - \frac{1}{2}\|\sum_{i=1}^{M} \rho_i \tilde{\mathbf{v}}_i'^{(t)}\|^2 \\
&\quad + \frac{1}{2}\|\sum_{i=1}^{M} \rho_i (\nabla f_i(\bar{\boldsymbol{\omega}}'^{(t)}) - \nabla f_i(\boldsymbol{\omega}_i'^{(t)}))\|^2 \\
&\leq -\frac{1}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 - \frac{1}{2}\|\sum_{i=1}^{M} \rho_i \tilde{\mathbf{v}}_i'^{(t)}\|^2 + \frac{L^2}{2}\sum_{i=1}^{M} \rho_i \|(\bar{\boldsymbol{\omega}}'^{(t)} - \boldsymbol{\omega}_i'^{(t)})\|^2
\end{aligned} \tag{A.18}$$

Our approach to the proof utilizes a similar structure to the one found in Wan et al. (2021a), given that the global mask $\mathbf{m}_t$ remains consistent throughout the local training process. Let $t_c = \left\lfloor \frac{t}{E_l} \right\rfloor E_l$ be the start time of local training. Then, $\bar{\boldsymbol{\omega}}'^{(t)}$ and $\boldsymbol{\omega}_i'^{(t)}$ can be written as:

$$\bar{\boldsymbol{\omega}}'^{(t)} = \left(\bar{\boldsymbol{\omega}}'^{(t_c)} - \frac{1}{C}\sum_{i\in P_t}\sum_{k=t_c+1}^{t-1} \eta_k \tilde{\mathbf{v}}_i'^{(k)}\right) \odot \mathbf{m}_t, \tag{A.19}$$

$$\boldsymbol{\omega}_i'^{(t)} = \left(\bar{\boldsymbol{\omega}}'^{(t_c)} - \sum_{k=t_c+1}^{t-1} \eta_k \tilde{\mathbf{v}}_i'^{(k)}\right) \odot \mathbf{m}_t. \tag{A.20}$$

According to Lemma 3, Assumption 3, Definition 2, and Eqs. (50)-(54) in Wan et al. (2021a), while taking the expectation over P_t, we get:

$$\sum_{i=1}^{M} \rho_i \|(\bar{\boldsymbol{\omega}}'^{(t)} - \boldsymbol{\omega}_i'^{(t)})\|^2 \leq (\gamma - 1)\left[\frac{1}{C}\sum_{i=1}^{M}\sum_{k=t_c+1}^{t-1} \eta_k^2 \rho_i \sigma_i^2 + \gamma E_l \sum_{k=t_c+1}^{t-1} \eta_k^2 \|\sum_{i=1}^{M} \rho_i \mathbf{v}_i'^{(k)}\|^2\right]. \tag{A.21}$$

Considering the decreasing learning rate $\eta_k^2 \leq \eta_t$ with $t_c + 1 < k < t$ and substituting Eq. (A.21) into (A.18), we have:

$$\begin{aligned}
-\mathbb{E}[\langle \tilde{\mathbf{v}}'^{(t)}, \nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\rangle] &\leq \frac{(\gamma-1)L^2\eta_t}{2C}\sum_{i=1}^{M} \rho_i \sigma_i^2 \\
&\quad + \frac{(\gamma-1)\gamma E_l \eta_t L^2}{2}\sum_{k=t_c+1}^{t_c+E_l} \|\sum_{i=1}^{M} \rho_i \mathbf{v}_i'^{(k)}\|^2 - \frac{1}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 - \frac{1}{2}\|\sum_{i=1}^{M} \rho_i \tilde{\mathbf{v}}_i'^{(t)}\|^2.
\end{aligned} \tag{A.22}$$

Substituting (A.17) and (A.22) into (A.13) completes the proof. □

Then we finally comes to prove the Theorem 2.

Proof. Recall (A.8), we sum up the results of Lemmas 4 and 5:

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})] \leq \frac{(\gamma-1)L^2\eta_t^2 + \eta_t^2 L}{2C}\sum_{i=1}^{M}\rho_i\sigma_i^2 - \frac{\eta_t}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 + \frac{(\gamma-1)\gamma E_l^2\eta_t^2 L^2 G^2}{2} + \mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|]. \quad (A.23)$$

Considering $\eta_t \leq \frac{1}{tL}$ and Assumption 4, Eq. (A.23) can be expressed as:

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})] \leq \frac{(\gamma-1)L^2\eta_t^2 + \eta_t^2 L}{2C}\sum_{i=1}^{M}\rho_i\sigma_i^2 - \frac{\eta_t}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 + \frac{(\gamma-1)\gamma E_l^2\eta_t^2 L^2 G^2}{2} + \mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|]. \quad (A.24)$$

Denoting $\chi = \frac{(\gamma-1)L^2+L}{2C}\sum_{i=1}^{M}\rho_i\sigma_i^2 + \frac{(\gamma-1)\gamma E_l^2 L^2 G^2}{2}$, Eq. (A.24) can be written as:

$$\frac{\eta_t}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 \leq \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] + \chi\eta_t^2 + \mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|]. \quad (A.25)$$

Taking the average of (A.25) over time T and rearranging, we obtain the following result, assuming that the model will converge to a stable point regarded as the optimum f^*:

$$\frac{1}{T}\sum_{t=1}^{T}\frac{1}{tL}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 \leq \frac{2}{T}\mathbb{E}(f(\boldsymbol{\omega}_1) - f^*) + \frac{2}{T}\sum_{t=1}^{T}[\mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|] + \chi\eta_t^2]. \quad (A.26)$$

Since $\min \frac{1}{T}\sum_{t=1}^{T}\frac{1}{T}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 \leq \frac{1}{T}\sum_{t=1}^{T}\frac{1}{t}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2$, Eq. (A.26) can be expressed as:

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 \leq 2L\mathbb{E}(f(\boldsymbol{\omega}_1) - f^*) + 2L\sum_{t=1}^{T}[\mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|] + \frac{\pi^2}{3L^2}\chi], \quad (A.27)$$

where it is known that $\sum_{t=1}^{T}\frac{1}{t^2} = \frac{\pi^2}{6}$. □

A.3.2 Proof for Corollary 1

Proof. Based on Equation (A.23), if the learning rate η is constant, then it is expressed as:

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})] \leq \frac{(\gamma-1)L^2\eta^2 + \eta^2 L}{2C}\sum_{i=1}^{M}\rho_i\sigma_i^2 - \frac{\eta}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 + \frac{(\gamma-1)\gamma E_l^2\eta^2L^2G^2}{2} + \mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|]. \quad \text{(A.28)}$$

Using the same logic in Theorem 2, after averaging and rearranging, Equation (A.26) can be reformulated as:

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 \leq \frac{2}{T\eta}\mathbb{E}(f(\boldsymbol{\omega}_1) - f^*) + \frac{2}{T\eta}\sum_{t=1}^{T}[\mu\mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|]] + 2\chi\eta. \quad \text{(A.29)}$$

Since $\min\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 \leq \frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2$, one can choose $\delta_t = \mathrm{o}(\frac{1}{T})$ which is faster than the chosen $\eta = \mathcal{O}(\frac{1}{T})$ (Lin et al., 2020), then $\min\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2$ converges at the speed of $\mathcal{O}(\frac{1}{T})$. □

A.4 Proof for Chapter 6

A.4.1 Auxiliary Lemmas

Firstly, note that in a time-varying connected topology, both $\mathcal{G}_i^t$ and $\mathcal{N}_i^t$ are randomly generated. For the purposes of theoretical analysis, we assume $\mathcal{N}_i^t = \mathcal{G}_i^t$, as our scheduling policy functions as a type of client selection policy.

Proposition 1. *Assume $\mathcal{M} = \{1, 2, \ldots, M\}$ clients. Then, exactly c neighbors in $\mathcal{N}_i^t$ have a reuse index less than i, following a hypergeometric distribution:*

$$\mathbb{P}(c, i) = \mathbb{P}(|\mathcal{N}_{(a)i}| = c) = \frac{\binom{i-1}{c}\binom{M-i}{C-c}}{\binom{M-1}{C}}, \quad \text{(A.30)}$$

where $c < C$ and $|\mathcal{N}_{(a)i}|$ is a subset of $\mathcal{N}_i^t$ with an index less than i.

Proof. Consider a system with i clients indexed from 1 to i, and let i be a particular client. We have:

1. There are $M-1$ potential clients to select from.
2. Among these $M-1$ clients, $m-1$ clients have an index less than i.
3. We aim to select C clients as neighbors of client i in each sample.

Thus, $|\mathcal{N}_{(a)i}|$ is a hypergeometric random variable. The probability $\mathbb{P}(c,i) = \mathbb{P}(|\mathcal{N}_{(a)i}| = c)$ that exactly c of the selected clients have an index less than i is given by

$$\mathbb{P}(c,i) = \frac{\binom{i-1}{c}\binom{M-i}{C-c}}{\binom{M-1}{C}}, \tag{A.31}$$

where $\sum_{0\leq c\leq \min(C,i-1)} \mathbb{P}(c,i) = 1$, which follows from Vandermonde's identity. □

After introducing the property of reusing index as a client selection policy, we review the local updates mechanism. The local update follows as in FedAvg:

$$\tilde{\boldsymbol{\omega}}_{k,\tau+1}^t = \tilde{\boldsymbol{\omega}}_{k,\tau}^t - \eta \mathbf{g}_{k,\tau}^t \odot \mathbf{m}_k^t, \tag{A.32}$$

where $\mathbf{g}_{k,\tau}^t = \nabla F_k(\tilde{\boldsymbol{\omega}}_{k,\tau}^t)$. This implies that

$$\eta \sum_{\tau=0}^{E_l-1} \mathbf{g}_{k,\tau}^t \odot \mathbf{m}_k^t = (\tilde{\boldsymbol{\omega}}_{k,0}^t - \tilde{\boldsymbol{\omega}}_{k,E_l}^t) \odot \mathbf{m}_k^t. \tag{A.33}$$

Note that $\boldsymbol{\omega}_k^{t+1} = \tilde{\boldsymbol{\omega}}_{k,E_l}^t$ and $\tilde{\boldsymbol{\omega}}_k^t = \tilde{\boldsymbol{\omega}}_{k,0}^t$. Hence, we first derive the upper bound for the local updates, as in Lemma 6.

Lemma 6. *Under Assumptions 2 and 5, for some neighbor size $C > 1$ and η such that $\eta^2 \leq \frac{1}{12C\mu^2(C-1)(2C-1)}$,*

$$\frac{1}{M}\sum_{i=1}^M \mathbb{E}\|\boldsymbol{\omega}_i^{t+1} - \tilde{\boldsymbol{\omega}}_i^t\|^2 \leq \left(e^{\frac{E_l}{2C-2}} - 1\right)(2C-2)\left(\frac{2C}{2C-1}\eta^2\sigma_l^2 + 6C\eta^2\sigma_g^2 + 6C\eta^2\frac{\sum_{i=1}^M \mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^t)\|^2}{M}\right). \tag{A.34}$$

Proof. Since the mask $\mathbf{m}_i^t$ remains consistent during training, the expression involving the corresponding model is omitted for brevity. Initially, consider the traditional weighted average aggregation where

$$\begin{aligned}\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau+1}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 = \mathbb{E}\Big\|&\tilde{\boldsymbol{\omega}}_{i,\tau}^t - \tilde{\boldsymbol{\omega}}_i^t - \eta\big(g_{i,\tau}^t \odot \mathbf{m}_i^t \\ &- \nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^t) + \nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^t) - \nabla f(\tilde{\boldsymbol{\omega}}_i^t) + \nabla f(\tilde{\boldsymbol{\omega}}_i^t) \\ &- \nabla f_i(\tilde{\boldsymbol{\omega}}_i^t) + \nabla f_i(\tilde{\boldsymbol{\omega}}_i^t)\big)\Big\|^2.\end{aligned} \tag{A.35}$$

Define $a := \mathbb{E}\big\|\tilde{\boldsymbol{\omega}}_{i,\tau}^t - \eta\big(\tilde{g}_{i,\tau}^t \odot \mathbf{m}_i^t - \nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^t)\big) - \tilde{\boldsymbol{\omega}}_i^t\big\|^2$ and $b = \eta^2\mathbb{E}\|\nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^t) - \nabla f(\tilde{\boldsymbol{\omega}}_i^t) + \nabla f(\tilde{\boldsymbol{\omega}}_i^t) - \nabla f_i(\tilde{\boldsymbol{\omega}}_i^t) + \nabla f_i(\tilde{\boldsymbol{\omega}}_i^t)\|^2$. Using Cauchy's inequality with an elastic variable $2C > 1$,

it follows that

$$\mathbb{E}\|\boldsymbol{\omega}_i^{t+1} - \tilde{\boldsymbol{\omega}}_i^t\|^2 \leq (1 + \frac{1}{2C-1})a + 2Cb. \tag{A.36}$$

By Assumptions 2 to 5 and the triangle inequality,

$$\begin{aligned} a &\leq \mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 + \eta^2\mathbb{E}\|\tilde{g}_{i,\tau}^t \odot \mathbf{m}_i^t - \nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^t)\|^2 \\ &= \mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 + \eta^2\sigma_l^2, \end{aligned} \tag{A.37}$$

and

$$\begin{aligned} b &\leq 3\eta^2\big[\mathbb{E}\|\nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^t) - \nabla f_i(\tilde{\boldsymbol{\omega}}_i^t)\|^2 + \mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^t)\|^2 + \mathbb{E}\|\nabla f_i(\tilde{\boldsymbol{\omega}}_i^t) - \nabla f(\tilde{\boldsymbol{\omega}}_i^t)\|^2\big] \\ &\leq 3\eta^2\big[\mathbb{E}\|\nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^t) - \nabla f_i(\tilde{\boldsymbol{\omega}}_i^t)\|^2 + \mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^t)\|^2 + \mathbb{E}\|\nabla f_i(\tilde{\boldsymbol{\omega}}_i^t) - \nabla f(\tilde{\boldsymbol{\omega}}^t)\|^2 \\ &\quad + \mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^t) - \nabla f(\tilde{\boldsymbol{\omega}}^t)\|^2\big] \\ &\leq 3\eta^2\big[\mu^2\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 + \mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^t)\|^2 + \sigma_g^2\big]. \end{aligned} \tag{A.38}$$

Substituting Eqs.(A.37) and (A.38) into Eq.(A.36) with some η such that $\eta^2 \leq \frac{1}{12C\mu^2(C-1)(2C-1)}$, yields

$$\begin{aligned} \mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau+1}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 &\leq (1 + \frac{1}{2C-1} + 6C\eta^2\mu^2)\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 + 6C\eta^2(\sigma_g^2 + \mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^t)\|^2) \\ &\quad + (1 + \frac{1}{2C-1})\eta^2\sigma_l^2 \\ &\leq (1 + \frac{1}{2C-2})\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 + (1 + \frac{1}{2C-1})\eta^2\sigma_l^2 + 6C\eta^2(\sigma_g^2 + \mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^t)\|^2) \end{aligned} \tag{A.39}$$

Let $D = 1 + \frac{1}{2(C-1)}$, $E = \left(1 + \frac{1}{2C-1}\right)\eta^2\sigma_l^2 + 6C\eta^2\sigma_g^2$, and $F = 6C\eta^2\mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^t)\|^2$, then the recursive inequality Eq.(A.39) becomes

$$\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau+1}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 \leq D\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,\tau}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 + E + F. \tag{A.40}$$

When $\tau = 0$, the initial condition is $\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,0}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 = 0$. For $\tau = 1$ to E_l, applying the inequality Eq.(A.40) E_l times, summing up the constants multiplied by their respective powers of D gives

$$\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,E_l}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 \leq D^{E_l}\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,0}^t - \tilde{\boldsymbol{\omega}}_i^t\|^2 + E\sum_{j=0}^{E_l-1} D^j + F\sum_{j=0}^{E_l-1} D^j. \tag{A.41}$$

The sums of the series can be simplified by the sum of a geometric series as follows

$$\sum_{j=0}^{E_l-1} D^j = \frac{1 - D^{E_l}}{1 - D}. \tag{A.42}$$

Hence the inequality can be further simplified as

$$\mathbb{E}\|\tilde{\boldsymbol{\omega}}_{i,E_l}^{t} - \tilde{\boldsymbol{\omega}}_{i}^{t}\|^2 \leq 0 + (E+F)\frac{D^{E_l}-1}{D-1}. \tag{A.43}$$

When $C > 1$, $D = 1 + \frac{1}{2C-2} < e^{\frac{1}{2C-2}}$ hence $D^{E_l} < e^{\frac{E_l}{2C-2}}$, which provides the final bound for $\mathbb{E}\|\boldsymbol{\omega}_i^{t+1} - \tilde{\boldsymbol{\omega}}_i^t\|^2$ as in Eq.(A.34). □

Then, before proving the theorem, we give the next lemma to see the contribution of the proposed dynamic *aggregation* policy.

Lemma 7. *Consider the proposed scheduling strategy. Let $\tilde{\boldsymbol{\omega}}_i^{t(\dagger)}$ denote the local personalized aggregated model for the i-th client at time t. The global aggregated model at time t, $\tilde{\boldsymbol{\omega}}^{t(\dagger)}$, is defined as the average of the local models, i.e., $\tilde{\boldsymbol{\omega}}^{t(\dagger)} = \frac{1}{M}\sum_{i=1}^{M}\tilde{\boldsymbol{\omega}}_i^{t(\dagger)}$, where i is the total number of clients. Let C represent the number of clients in the neighborhood. With the support of Lemma 1 in Wan et al. (2021b), the expected value of the global model at time $t+1$, denoted as $\mathbb{E}(\tilde{\boldsymbol{\omega}}^{t+1(\dagger)})$, is given by*

$$\mathbb{E}(\tilde{\boldsymbol{\omega}}^{t+1(\dagger)}) = \mathbb{E}(\tilde{\boldsymbol{\omega}}^{t(\dagger)}) - \eta\frac{\mathbb{E}\left(\sum_{i=1}^{M}\sum_{\tau=0}^{E_l^*-1}\mathbf{g}_{\tau,i}(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\right)}{M}, \tag{A.44}$$

where $E_l^* = \frac{3C+2}{2(C+1)}E_l$, and E_l is the number of steps of local updates. Here, $\mathbf{g}_{\tau,i}$ represents the gradient computation for the i-th client at local update step τ.

Proof. Consider the effect of the proposed dynamic aggregation policy. Rewrite the mask element aggregation with the sequential appointment as

$$\tilde{\boldsymbol{\omega}}_i^{t(\dagger)} = \left(\frac{\sum_{j\in\mathcal{N}_{(a)i}^t}\boldsymbol{\omega}_j^t + \sum_{j\in\mathcal{N}_{(b)i}^t}\boldsymbol{\omega}_j^t + \boldsymbol{\omega}_i^t}{\sum_{j\in\mathcal{N}_{(a)i}^t}\mathbf{m}_j^t + \sum_{j\in\mathcal{N}_{(b)i}^t}\mathbf{m}_j^t + \mathbf{m}_i^t}\right)\odot\mathbf{m}_i^t \tag{A.45}$$

$$= \left(\frac{\sum_{j\in\mathcal{N}_{(a)i}^t}\boldsymbol{\omega}_j^t + \sum_{j\in\mathcal{N}_{(b)i+}^t}\boldsymbol{\omega}_j^t}{C+1}\right)\odot\mathbf{m}_i^t, \tag{A.46}$$

where $\mathcal{N}_{(b)i}^t := \mathcal{N}_i^t \setminus \mathcal{N}_{(a)i}^t$, $\mathcal{N}_{(b)i+}^t := \mathcal{N}_{(b)i}^t \cup \{m\}$, and the last equation holds under Assumption 6.

Similar to Eq.(A.33), the term $\mathbf{m}_i^t$ is omitted for convenience since the mask is consistent during local training. For the j-th client at time t, the local personalized aggregated model is

$$\boldsymbol{\omega}_j^{t(\dagger)} = \begin{cases}\tilde{\boldsymbol{\omega}}_j^t - \eta\sum_{\tau=0}^{E_l-1}\mathbf{g}_{j,\tau}^t\odot\mathbf{m}_j^t, & \text{if } j\in\mathcal{N}_{(a)i}^t;\\ \boldsymbol{\omega}_j^t, & \text{otherwise.}\end{cases} \tag{A.47}$$

When $j \in \mathcal{N}_{(a)i}^t$, $\boldsymbol{\omega}_j^{t(\dagger)}$ is equivalent to $\boldsymbol{\omega}_j^{t+1}$, reflecting the scenario where all participating clients perform an equal number of local training iterations within a single commu-

nication round, analogous to traditional FL. The superscript (†) explicitly signifies that although the local gradients $\mathbf{g}_{j,\tau}^t$ are computed under varying aggregation models, they are distinct from those in a parallel FL framework. Denote $I_{\{j\in\mathcal{N}_{(a)i}^t\}}$ as an indicator function for the event that the j-th client is selected in the delayed neighborhoods of client i.

Using the results of Proposition 1, it can be shown that

$$\begin{aligned}
\mathbb{E}(\sum_{j\in\mathcal{N}_{(a)i}^t}\boldsymbol{\omega}_j^{t+1}) &= \mathbb{E}(\mathbb{E}(\sum_{j\in\mathcal{N}_{(a)i}^t}\boldsymbol{\omega}_j^{t+1}||\mathcal{N}_{(a)i}^t|)) = \mathbb{E}(\mathbb{E}(\sum_{j\in\mathcal{N}_i^t}\boldsymbol{\omega}_j^{t+1}I_{\{j\in\mathcal{N}_{(a)i}^t\}}||\mathcal{N}_{(a)i}^t|)) \\
&= \mathbb{E}(\mathbb{E}(\sum_{j\in\mathcal{N}_i^t}\boldsymbol{\omega}_j^{t+1}I_{\{j\in\mathcal{N}_{(a)i}^t||\mathcal{N}_{(a)i}^t|\}})) = \mathbb{E}(\sum_{j\in\mathcal{N}_i^t}\boldsymbol{\omega}_j^{t+1}\mathbb{E}(I_{\{j\in\mathcal{N}_{(a)i}^t||\mathcal{N}_{(a)i}^t|\}})) \\
&= \mathbb{E}(\sum_{j\in\mathcal{N}_i^t}\boldsymbol{\omega}_j^{t+1}\mathbb{P}(j\in\mathcal{N}_{(a)i}^t||\mathcal{N}_{(a)i}^t|)) = \mathbb{E}(\sum_{j\in\mathcal{N}_i^t}\boldsymbol{\omega}_j^{t+1}\frac{|\mathcal{N}_{(a)i}^t|}{C}) \\
&= \mathbb{E}(|\mathcal{N}_{(a)i}^t|)\frac{\mathbb{E}(\sum_{j\in\mathcal{N}_i^t}\boldsymbol{\omega}_j^{t+1})}{C} = \frac{(i-1)C}{M-1}\mathbb{E}(\bar{\boldsymbol{\omega}}^{t+1}),
\end{aligned} \tag{A.48}$$

where $|\mathcal{N}_{(a)i}^t|$ follows a hypergeometric distribution. Similarly,

$$\begin{aligned}
\mathbb{E}(\sum_{j\in\mathcal{N}_{(b)i+}^t}\boldsymbol{\omega}_j^t) &= \mathbb{E}(\mathbb{E}(\sum_{j\in\mathcal{N}_{(b)i+}^t}\boldsymbol{\omega}_j^t||\mathcal{N}_{(b)i+}^t|)) = \mathbb{E}(\mathbb{E}(\sum_{j\in\mathcal{N}_{i+}^t}\boldsymbol{\omega}_j^tI_{\{j\in\mathcal{N}_{(b)i+}^t\}}||\mathcal{N}_{(b)i+}^t|)) \\
&= \mathbb{E}(\mathbb{E}(\sum_{j\in\mathcal{N}_{i+}^t}\boldsymbol{\omega}_j^tI_{\{j\in\mathcal{N}_{(b)i+}^t||\mathcal{N}_{(b)i+}^t|\}})) = \mathbb{E}(\sum_{j\in\mathcal{N}_{i+}^t}\boldsymbol{\omega}_j^t\mathbb{E}(I_{\{j\in\mathcal{N}_{(b)i+}^t||\mathcal{N}_{(b)i+}^t|\}})) \\
&= \mathbb{E}(\sum_{j\in\mathcal{N}_{i+}^t}\boldsymbol{\omega}_j^t\mathbb{P}(j\in\mathcal{N}_{(b)i+}^t||\mathcal{N}_{(b)i+}^t|)) = \mathbb{E}(\sum_{j\in\mathcal{N}_{i+}^t}\boldsymbol{\omega}_j^t\frac{|\mathcal{N}_{(b)i+}^t|-1}{C}) \\
&= \mathbb{E}((|\mathcal{N}_{(b)i+}^t|-1))\frac{\mathbb{E}(\sum_{j\in\mathcal{N}_{i+}^t}\boldsymbol{\omega}_j^t)}{C} = \frac{(M-i)(C+1)}{(M-1)}\mathcal{E}(\bar{\boldsymbol{\omega}}^t).
\end{aligned} \tag{A.49}$$

Therefore, $\mathbb{E}(\tilde{\boldsymbol{\omega}}_i^{t(\dagger)})$ can be written as

$$\begin{aligned}
&[\frac{(i-1)C}{(M-1)(C+1)}\mathbb{E}(\bar{\boldsymbol{\omega}}^{t+1}) + \frac{(M-i)}{(M-1)}\mathbb{E}(\bar{\boldsymbol{\omega}}^t)]\odot\mathbf{m}_i^t \\
&= \mathbb{E}(\bar{\boldsymbol{\omega}}^t)\odot\mathbf{m}_i^t - [\frac{(i-1)C}{(M-1)(C+1)}\mathbb{E}(\frac{\eta\sum_{j\in\mathcal{N}_{i+}^t}\sum_{\tau=0}^{E_l-1}g_{j,\tau}^t}{C+1})]\odot\mathbf{m}_i^t,
\end{aligned} \tag{A.50}$$

where one can verify that when $i=1$, $\mathbb{E}(\tilde{\boldsymbol{\omega}}_i^{t(\dagger)})$ reduces to $\mathbb{E}(\tilde{\boldsymbol{\omega}}_i^t)$. Recall that $\tilde{\boldsymbol{\omega}}^{t(\dagger)} =$

$\frac{1}{M}\sum_{i=1}^{M}\tilde{\boldsymbol{\omega}}_i^{t(\dagger)}$, then

$$\begin{aligned}\mathbb{E}(\tilde{\boldsymbol{\omega}}^{t(\dagger)}) &= \frac{1}{M}\sum_{i=1}^{M}(\mathbb{E}(\bar{\boldsymbol{\omega}}^t - \frac{(i-1)C}{(M-1)(C+1)}\eta\sum_{j\in\mathcal{N}_{i+}^t}\sum_{\tau=0}^{E_l-1} g_{j,\tau}^t(\bar{\boldsymbol{\omega}}^t))\odot\mathbf{m}_i^t)\\ &= \frac{1}{M}\sum_{i=1}^{M}(\mathbb{E}(\bar{\boldsymbol{\omega}}^t) - \frac{(i-1)C}{(M-1)(C+1)}\eta\mathbb{E}(\tilde{g}_i^t)\odot\mathbf{m}_i^t) \qquad (A.51)\\ &= \mathbb{E}(\bar{\boldsymbol{\omega}}^t) - \frac{1}{M}\sum_{i=1}^{M}(\frac{(i-1)C}{(M-1)(C+1)}\eta\mathbb{E}(\tilde{g}_i^t))\odot\mathbf{m}_i^t, \qquad (A.52)\end{aligned}$$

where $\tilde{g}_i^t = \frac{\sum_{j\in\mathcal{N}_{i+}^t}\sum_{\tau=0}^{E_l-1} g_{j,\tau}^t}{C+1}$ and the last equality holds according to the definition of $\bar{\boldsymbol{\omega}}^t$. $g_{j,\tau}^t(\bar{\boldsymbol{\omega}})$ indicates the gradient at local epoch $\tau=0$ is with respect to $\bar{\boldsymbol{\omega}}$.

Let $\tilde{g}^{t(\dagger)} = \frac{1}{M}\sum_{i=1}^{M}[\frac{(i-1)C}{(M-1)(C+1)}\tilde{g}_i^t\odot\mathbf{m}_i^t]$, then

$$\mathbb{E}(\tilde{\boldsymbol{\omega}}^{t(\dagger)}) = \mathbb{E}(\bar{\boldsymbol{\omega}}^t) - \eta\mathbb{E}(\tilde{g}^{t(\dagger)}). \qquad (A.53)$$

To find the boundary for the difference between the global model at time t and $t+1$, and according to Assumption 2, we have

$$\begin{aligned}&\mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{t+1(\dagger)})] - \mathcal{E}[f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})]\\ &\leq \mathbb{E}\left[\left\langle f(\nabla\tilde{\boldsymbol{\omega}}^{t(\dagger)}), \tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)}\right\rangle\right]\\ &\quad + \frac{\mu}{2}\|\tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)}\|^2. \qquad (A.54)\end{aligned}$$

Lemma 1 in (Wan et al., 2021b) ensures that $\forall m\in|M|,\ \mathbb{E}\left(\frac{\sum_{i=1}^{M}\sum_{\tau=0}^{E_l-1}\mathbf{g}_{\tau,i}(\tilde{\boldsymbol{\omega}}^{t(\dagger)})}{M}\right) = \mathbb{E}[\tilde{\mathbf{g}}_i^t]$ since $\mathcal{N}_i^t$ is selected randomly. According to the definition and Eq. (A.53),

$$\begin{aligned}\mathbb{E}(\tilde{\boldsymbol{\omega}}^{t+1(\dagger)}) &= \mathbb{E}(\bar{\boldsymbol{\omega}}^{t+1}) - \eta\mathbb{E}(\tilde{g}^{t+1(\dagger)})\\ &= \mathbb{E}(\tilde{\boldsymbol{\omega}}^{t(\dagger)}) - \eta\frac{\mathbb{E}\left(\sum_{i=1}^{M}\sum_{\tau=0}^{E_l-1}\mathbf{g}_{\tau,i}(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\right)}{M} - \eta\mathbb{E}(\tilde{g}^{t+1(\dagger)}) \qquad (A.55)\\ &= \mathbb{E}(\tilde{\boldsymbol{\omega}}^{t(\dagger)}) - \eta\frac{\mathbb{E}\left(\sum_{i=1}^{M}\sum_{\tau=0}^{E_l^*-1}\mathbf{g}_{\tau,i}(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\right)}{M}, \qquad (A.56)\end{aligned}$$

where $E_l^* = E_l + \frac{C}{2(C+1)}E_l$. Here, the starting weights of $\tilde{g}^{t+1(\dagger)}$ are with respect to $\mathbb{E}(\bar{\boldsymbol{\omega}}^{t+1}) = \mathbb{E}(\tilde{\boldsymbol{\omega}}^{t(\dagger)}) - \eta\frac{\sum_{i=1}^{M}\mathbb{E}\sum_{\tau=0}^{E_l-1}(\mathbf{g}_{\tau,i}(\tilde{\boldsymbol{\omega}}^{t(\dagger)}))}{M}$. One can think of just continuing $\frac{C}{2(C+1)}$ more steps of local training, where $\frac{1}{M}\sum_{i=1}^{M}[\frac{(i-1)C}{(M-1)(C+1)}] = \frac{C}{2(C+1)}$. E_l^* might not be an integer, but this concludes the effect of sequential aligning for convergence analysis. □

Our objective is now to establish the bounds for $\mathbb{E}\langle f(\nabla\tilde{\boldsymbol{\omega}}^{t(\dagger)}), \tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)}\rangle$ and

$\frac{\mu}{2}\|\tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)}\|^2$.

Lemma 8. *Under Assumptions 2 to 5 and Lemma 7, suppose $\tilde{\boldsymbol{\omega}}^{t+1(\dagger)}$ and $\tilde{\boldsymbol{\omega}}^{t(\dagger)}$ are global models learned by the proposed strategy. For some $C > 1$ with $\eta \leq \sqrt{\frac{1}{12C\mu^2(C-1)(2C-1)}}$, $\mathbb{E}\left[\frac{\mu}{2}\|\tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)}\|^2\right]$ is upper bounded by*

$$\mathbb{E}\left[\frac{\mu}{2}\|\tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)}\|^2\right] \leq \mu S_1(S_2 + 3\mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2),$$

where $S_1 = 2\eta^2 C(C-1)\left(\exp\left(\frac{(3C+2)E_l}{4(C^2-1)}\right) - 1\right)$ and $S_2 = \frac{1}{2C-1}\sigma_l^2 + 3(2\sigma_g^2 + \sigma_p^2)$.

Proof. Lemma 6 provides the boundary for the local updates. Equation (A.55) and Lemma 7 indicate that the scheduling increases the local epochs E_l to E_l^* with the expectation for the (aggregated) global model. Hence,

$$\begin{aligned}\mathbb{E}\left[\|\tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)}\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{M}\sum_{i=1}^{M}\tilde{\boldsymbol{\omega}}_i^{t+1(\dagger)} - \frac{1}{M}\sum_{i=1}^{M}\tilde{\boldsymbol{\omega}}_i^{t(\dagger)}\right\|^2\right] \\ &\leq \mathbb{E}\left[\frac{1}{M}\sum_{i=1}^{M}\|\tilde{\boldsymbol{\omega}}_i^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}_i^{t(\dagger)}\|^2\right]. \end{aligned} \tag{A.57}$$

The last inequality holds due to Jensen's Inequality where the defined function $\phi(\cdot) = \|\cdot\|^2$ is convex.

Therefore, substituting the results of Lemma 6 and replacing E_l with $E_l^* + 1$, then the boundary with $\frac{\sum_{i=1}^{M}\mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^{t(\dagger)})\|^2}{M}$ is remained to be proved. Using the triangle inequality again and Assumption 5,

$$\begin{aligned}\frac{\sum_{i=1}^{M}\mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}_i^{t(\dagger)})\|^2}{M} &\leq \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}\Big(\|\nabla f_i(\tilde{\boldsymbol{\omega}}_i^{t(\dagger)}) - \nabla f(\tilde{\boldsymbol{\omega}}_i^{t(\dagger)})\|^2 \\ &\quad + \|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2 \\ &\quad + \|\nabla f_i(\tilde{\boldsymbol{\omega}}_i^{t(\dagger)}) - \nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2\Big) \\ &\leq \mathbb{E}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2 + \sigma_p^2 + \sigma_g^2. \end{aligned} \tag{A.58}$$

□

Lemma 9. *Under Assumptions 2 to 5 and Lemma 8, $\mathbb{E}\left[\left\langle\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)}), \tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)}\right\rangle\right]$ is upper bounded by*

$$-\frac{\eta}{2}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2 - \frac{\eta}{2}\mathbb{E}\|\hat{\mathbf{g}}^{t(\dagger)}\|^2 + \frac{3\mu^2\eta^3 E_l^*}{C}(\sigma_l^2 + \sigma_g^2), \tag{A.59}$$

where $E_l^* = \frac{3C+2}{2(C+1)}E_l$.

Proof. According to Eq. (A.55), let $\hat{\mathbf{g}}^{t(\dagger)} = \frac{\sum_{i=1}^{M}\sum_{\tau=0}^{E_l^*}\mathbf{g}_{\tau,i}(\tilde{\boldsymbol{\omega}}^{t(\dagger)})}{M}$,

$$\mathbb{E}\left[\left\langle \nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)}), \tilde{\boldsymbol{\omega}}^{t+1(\dagger)} - \tilde{\boldsymbol{\omega}}^{t(\dagger)} \right\rangle\right] = -\eta\mathbb{E}\left[\left\langle \nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)}), \mathbb{E}\left(\hat{\mathbf{g}}^{t(\dagger)}\right)\right\rangle\right]. \tag{A.60}$$

To bound $-\mathbb{E}\left[\left\langle \nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)}), \mathbb{E}\left(\hat{\mathbf{g}}^{t(\dagger)}\right)\right\rangle\right]$,

$$\begin{aligned}
&-\mathbb{E}\left[\left\langle \nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)}), \mathbb{E}\left(\hat{\mathbf{g}}^{t(\dagger)}\right)\right\rangle\right] \\
&= -\frac{1}{2}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2 - \frac{1}{2}\mathbb{E}\|\hat{\mathbf{g}}^{t(\dagger)}\|^2 + \frac{1}{2}\left\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)}) - \mathbb{E}[\hat{\mathbf{g}}^{t(\dagger)}]\right\|^2,
\end{aligned} \tag{A.61}$$

where the equality follows from the identity $\langle \mathbf{a}, \mathbf{b}\rangle = \frac{\|\mathbf{a}\|^2+\|\mathbf{b}\|^2-\|\mathbf{a}-\mathbf{b}\|^2}{2}$. The bounds for the first two terms are established in Lemma 8. For the third term, using Assumption 2 and denoting t_c as the start of the communication round of t:

$$\begin{aligned}
&\frac{1}{2}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)}) - \mathbb{E}[\hat{\mathbf{g}}^{t(\dagger)}]\|^2 \leq \frac{\mu^2}{2M}\sum_{i=1}^{M}\mathbb{E}\|\tilde{\boldsymbol{\omega}}^{t(\dagger)} - \tilde{\boldsymbol{\omega}}_i^{t(\dagger)}\|^2 \\
&= \frac{\mu^2}{2}\sum_{i=1}^{M}\mathbb{E}\Bigg[\frac{1}{M}\Bigg\|\tilde{\boldsymbol{\omega}}^{t_c(\dagger)} - \frac{1}{M}\sum_{j=1}^{M}\mathbb{E}_{\mathcal{P}_{t,j}}\left(\frac{\eta}{C}\sum_{j\in\mathcal{P}_{t,j}}\sum_{\tau=t_c}^{t_c+E_l^*}\tilde{g}_{\tau,j}^{t_c(\dagger)}\right) \\
&\quad - \tilde{\boldsymbol{\omega}}^{t_c(\dagger)} + \mathbb{E}_{\mathcal{P}_{t,i}}\left(\frac{\eta}{C}\sum_{i\in\mathcal{P}_{t,i}}\sum_{\tau=t_c}^{t_c+E_l^*}\tilde{g}_{\tau,i}^{t_c(\dagger)}\right)\Bigg\|^2\Bigg] \\
&= \frac{\mu^2\eta^2}{2}\sum_{i=1}^{M}\mathbb{E}\Bigg[\frac{1}{M}\Bigg\|\mathbb{E}_{\mathcal{P}_{t,i}}\left(\frac{1}{C}\sum_{i\in\mathcal{P}_{t,i}}\sum_{\tau=t_c}^{t_c+E_l^*}\tilde{g}_{\tau,i}^{t_c(\dagger)}\right) \\
&\quad - \frac{1}{M}\sum_{j=1}^{M}\mathbb{E}_{\mathcal{P}_{t,j}}\left(\frac{1}{C}\sum_{j\in\mathcal{P}_{t,j}}\sum_{\tau=t_c}^{t_c+E_l^*}\tilde{g}_{\tau,j}^{t_c(\dagger)}\right)\Bigg\|^2\Bigg] \\
&= \frac{\mu^2\eta^2}{2}\sum_{i=1}^{M}\mathbb{E}\Bigg[\frac{1}{M}\Bigg\|\mathbb{E}_{\mathcal{P}_{t,i}}\left(\frac{1}{C}\sum_{i\in\mathcal{P}_{t,i}}\sum_{\tau=t_c}^{t_c+E_l^*}(\tilde{g}_{\tau,i}^{t_c(\dagger)} - \nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^{t_c(\dagger)}))\right) \\
&\quad - \frac{1}{M}\sum_{j=1}^{M}\mathbb{E}_{\mathcal{P}_{t,j}}\left(\frac{1}{C}\sum_{j\in\mathcal{P}_{t,j}}\sum_{\tau=t_c}^{t_c+E_l^*}(\tilde{g}_{\tau,j}^{t_c(\dagger)} - \nabla f_j(\tilde{\boldsymbol{\omega}}_{j,\tau}^{t_c(\dagger)}))\right) \\
&\quad + \mathbb{E}_{\mathcal{P}_{t,i}}\left(\frac{1}{C}\sum_{i\in\mathcal{P}_{t,i}}\sum_{\tau=t_c}^{t_c+E_l^*}\nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^{t_c(\dagger)})\right) \\
&\quad - \frac{1}{M}\sum_{j=1}^{M}\mathbb{E}_{\mathcal{P}_{t,j}}\left(\frac{1}{C}\sum_{j\in\mathcal{P}_{t,j}}\sum_{\tau=t_c}^{t_c+E_l^*}\nabla f_j(\tilde{\boldsymbol{\omega}}_{j,\tau}^{t_c(\dagger)})\right)\Bigg\|^2\Bigg]
\end{aligned}$$

$$
\begin{aligned}
&\overset{(b)}{\leq} \frac{3\mu^2\eta^2}{2M}\sum_{i=1}^{M}\Bigg[\mathbb{E}_{\mathcal{P}_{t,j}}\Bigg\|\frac{1}{C}\sum_{i\in\mathcal{P}_{t,i}}\sum_{\tau=t_c}^{t_c+E_l^*}(\tilde{g}_{\tau,i}^{t_c(\dagger)}-\nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^{t_c(\dagger)}))\Bigg\|^2 \\
&\quad+\frac{1}{M}\sum_{j=1}^{M}\mathbb{E}_{\mathcal{P}_{t,j}}\Bigg\|\frac{1}{C}\sum_{j\in\mathcal{P}_{t,j}}\sum_{\tau=t_c}^{t_c+E_l^*}\nabla f_j(\tilde{\boldsymbol{\omega}}_{j,\tau}^{t_c(\dagger)})\Bigg\|^2 \\
&\quad+\Bigg\|\mathbb{E}_{\mathcal{P}_{t,i}}\Bigg(\frac{1}{C}\sum_{i\in\mathcal{P}_{t,i}}\sum_{\tau=t_c}^{t_c+E_l^*}\nabla f_i(\tilde{\boldsymbol{\omega}}_{i,\tau}^{t_c(\dagger)})\Bigg) \\
&\quad-\frac{1}{M}\sum_{j=1}^{M}\mathbb{E}_{\mathcal{P}_{t,j}}\Bigg(\frac{1}{C}\sum_{j\in\mathcal{P}_{t,j}}\sum_{\tau=t_c}^{t_c+E_l^*}\nabla f_j(\tilde{\boldsymbol{\omega}}_{j,\tau}^{t_c(\dagger)})\Bigg)\Bigg\|^2\Bigg] \\
&\overset{(c)}{\leq} \frac{3\mu^2\eta^2}{2M}\left[\frac{E_l^*}{C}\sigma_l^2+\frac{E_l^*M\sigma_l^2}{CM}+\frac{E_l^*}{C}\sigma_g^2+\frac{E_l^*M\sigma_g^2}{CM}\right] \\
&= \frac{3\mu^2\eta^2E_l^*}{C}(\sigma_l^2+\sigma_g^2)
\end{aligned}
\tag{A.62}
$$

In the above formulation, the variable j serves to distinguish from i, ensuring clarity in the representation of individual client contributions. Here, $\mathcal{P}_{t,i}$ denotes the selection probability of client i at time t. The inequality marked as (b) derives from the application of the Cauchy-Schwarz inequality, exemplified by the relation $\|a+b+c\|^2 \leq 3(\|a\|^2 + \|b\|^2+\|c\|^2)$. The step labeled as (c) leverages a similar analytical technique, focusing on the aggregation of global gradients, thereby facilitating the derivation of σ_g^2, which is in conjunction with Assumption 5.

Substitute the results from Lemma 8 into Eq.(A.61) and multiply by η to conclude the proof for Lemma 9. □

A.4.2 Proof for Theorem 3

Proof. Combining the bounds from Lemma 8 and Lemma 5, the following is obtained:

$$
\begin{aligned}
\mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{t+1(\dagger)})]-\mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})] &\leq \mathbb{E}\left[\frac{\mu}{2}\|\tilde{\boldsymbol{\omega}}^{t+1(\dagger)}-\tilde{\boldsymbol{\omega}}^{t(\dagger)}\|^2\right]+\mathbb{E}[\langle\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)}),\tilde{\boldsymbol{\omega}}^{t+1(\dagger)}-\tilde{\boldsymbol{\omega}}^{t(\dagger)}\rangle] \\
&= -\frac{\eta}{2}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2+\frac{\mu-\eta}{2}\mathbb{E}\|\hat{\mathbf{g}}^{t(\dagger)}\|^2+\frac{3\mu^2\eta^3E_l^*}{C}(\sigma_l^2+\sigma_g^2) \qquad \text{(A.63)} \\
&\leq \frac{6S_1(\mu-\eta)-\eta}{2}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2+(\mu-\eta)S_1S_2+ \\
&\quad\frac{3\mu^2\eta^3\left(\frac{3C+2}{2(C+1)}E_l-1\right)}{C}(\sigma_l^2+\sigma_g^2). \qquad \text{(A.64)}
\end{aligned}
$$

It is a fact that $\min\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2 \leq \frac{\sum_{t=0}^{T-1}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2}{T}$. Summing up Eq. (A.63) from $t=0$ to a large $t=T$ concludes the proof. In detail, given the inequality for each iteration

t:

$$\begin{aligned}\mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{t+1(\dagger)})] - \mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})] \leq \frac{6S_1(\mu-\eta)-\eta}{2}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2 \\ + (\mu-\eta)S_1S_2 + \frac{3\mu^2\eta^3\left(\frac{3C+2}{2(C+1)}E_l - 1\right)}{C}(\sigma_l^2+\sigma_g^2).\end{aligned} \tag{A.65}$$

Summing this inequality from $t=0$ to $t=T-1$ yields:

$$\begin{aligned}\mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{T(\dagger)})] - \mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{0(\dagger)})] \leq \sum_{t=0}^{T-1}\left(\frac{6S_1(\mu-\eta)-\eta}{2}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2\right) \\ + T\cdot\left[(\mu-\eta)S_1S_2 + \frac{3\mu^2\eta^3\left(\frac{3C+2}{2(C+1)}E_l - 1\right)}{C}(\sigma_l^2+\sigma_g^2)\right]\end{aligned} \tag{A.66}$$

To isolate the cumulative gradient norm terms across T iterations, divide the inequality by the coefficient of the gradient norm term:

$$\sum_{t=0}^{T-1}\|\nabla f(\tilde{\boldsymbol{\omega}}^{t(\dagger)})\|^2 \leq \frac{2}{\eta - 6S_1(\mu-\eta)}\left(\mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{0(\dagger)})] - \mathbb{E}[f(\tilde{\boldsymbol{\omega}}^{T(\dagger)})]\right) + S_3T, \tag{A.67}$$

where $S_3 = \frac{2}{\eta-6S_1(\mu-\eta)}\cdot\left[(\mu-\eta)S_1S_2 + \frac{3\mu^2\eta^3(3C+2)E_l}{2(C+1)C}(\sigma_l^2+\sigma_g^2)\right]$. When T is large, the denominator $\eta - 6S_1(\mu-\eta)$ is dominated by η. Then, when $\eta \propto \mathcal{O}(\frac{1}{\sqrt{T}\mu})$ and as T is large enough, S_3 diminishes closely to 0. □

Bibliography

Abdi, A., Fekri, F., 2020. Quantized compressive sampling of stochastic gradients for efficient communication in distributed deep learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3105–3112.

Aji, A.F., Heafield, K., 2017. Sparse communication for distributed gradient descent, in: EMNLP'17, pp. 440–445.

Alistarh, D., Hoefler, T., Johansson, M., Konstantinov, N., Khirirat, S., Renggli, C., 2018. The convergence of sparsified gradient methods. NeurIPS'18 31.

Amiri, M.M., Gunduz, D., Kulkarni, S.R., Poor, H.V., 2020. Federated learning with quantized global model updates. arXiv preprint arXiv:2006.10672 .

Anagnostopoulos, C., 2020. Edge-centric inferential modeling & analytics. Journal of Network and Computer Applications 164, 102696. URL: https://www.sciencedirect.com/science/article/pii/S1084804520301703, doi:https://doi.org/10.1016/j.jnca.2020.102696.

Anagnostopoulos, C., Hadjiefthymiades, S., Zervas, E., 2011. Information dissemination between mobile nodes for collaborative context awareness. IEEE Transactions on Mobile Computing 10, 1710–1725. doi:10.1109/TMC.2011.19.

Anagnostopoulos, C., Kolomvatsos, K., 2018. Predictive intelligence to the edge through approximate collaborative context reasoning. Applied Intelligence 48. doi:10.1007/s10489-017-1032-y.

Badeau, R., David, B., Richard, G., 2005. Fast approximated power iteration subspace tracking. IEEE Transactions on Signal Processing 53, 2931–2941. doi:10.1109/TSP.2005.850378.

Bartlett, P.L., Montanari, A., Rakhlin, A., 2021. Deep learning: a statistical viewpoint. Acta numerica 30, 87–201.

Bellal, Z., Nour, B., Mastorakis, S., 2021. Coxnet: A computation reuse architecture at the edge. IEEE transactions on green communications and networking 5, 765–777.

Beltrán, E.T.M., Pérez, M.Q., Sánchez, P.M.S., Bernal, S.L., Bovet, G., Pérez, M.G., Pérez, G.M., Celdrán, A.H., 2023. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. IEEE Communications Surveys & Tutorials .

Bibikar, S., Vikalo, H., Wang, Z., Chen, X., 2022. Federated dynamic sparse training: Computing less, communicating less, yet learning better, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 6080–6088.

Bishop, C.M., 2007. Pattern Recognition and Machine Learning (Information Science and Statistics). 1 ed., Springer.

Blitzer, J., Dredze, M., Pereira, F., 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification, in: Proceedings of the 45th annual meeting of the association of computational linguistics, pp. 440–447.

Bottou, L., 2012. Stochastic gradient descent tricks, in: Neural networks: Tricks of the trade. Springer, pp. 421–436.

Boyd, S., Boyd, S.P., Vandenberghe, L., 2004. Convex optimization. Cambridge university press.

Cao, W., Wu, S., Yu, Z., Wong, H.S., 2018. Exploring correlations among tasks, clusters, and features for multitask clustering. IEEE transactions on neural networks and learning systems 30, 355–368.

Caruana, R., 1997. Multitask learning. Machine learning 28, 41–75.

Cavallanti, G., Cesa-Bianchi, N., Gentile, C., 2010. Linear algorithms for online multitask classification. The Journal of Machine Learning Research 11, 2901–2934.

Che, C., Li, X., Chen, C., He, X., Zheng, Z., 2022. A decentralized federated learning framework via committee mechanism with convergence guarantee. IEEE Transactions on Parallel and Distributed Systems 33, 4783–4800.

Chen, D., Yao, L., Gao, D., Ding, B., Li, Y., 2023. Efficient personalized federated learning via sparse model-adaptation, in: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (Eds.), Proceedings of the 40th International Conference on Machine Learning, PMLR. pp. 5234–5256. URL: https://proceedings.mlr.press/v202/chen23aj.html.

Chen, J., Ran, X., 2019. Deep learning with edge computing: A review. Proceedings of the IEEE 107, 1655–1674.

Chen, M., Shlezinger, N., Poor, H.V., Eldar, Y.C., Cui, S., 2021. Communication-efficient federated learning. Proceedings of the National Academy of Sciences 118, e2024789118.

Chen, T., Giannakis, G., Sun, T., Yin, W., 2018. Lag: Lazily aggregated gradient for communication-efficient distributed learning. NeurIPS'18 31.

Cheng, Y., Wang, D., Zhou, P., Zhang, T., 2017. A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282 .

Dai, R., Shen, L., He, F., Tian, X., Tao, D., 2022. Dispfl: Towards communication-efficient personalized federated learning via decentralized sparse training, in: International Conference on Machine Learning, PMLR. pp. 4587–4604.

Dehghani, M., Yazdanparast, Z., 2023. From distributed machine to distributed deep learning: a comprehensive survey. Journal of Big Data 10, 158.

Derakhshan, B., Rezaei Mahdiraji, A., Kaoudi, Z., Rabl, T., Markl, V., 2022. Materialization and reuse optimizations for production data science pipelines, in: Proceedings of the 2022 International Conference on Management of Data, Association for Computing Machinery, New York, NY, USA. p. 1962–1976. URL: https://doi.org/10.1145/3514221.3526186, doi:10.1145/3514221.3526186.

Diao, E., Wang, G., Zhang, J., Yang, Y., Ding, J., Tarokh, V., 2022. Pruning deep neural networks from a sparsity perspective, in: The Eleventh International Conference on Learning Representations.

Dice, L.R., 1945. Measures of the amount of ecologic association between species. Ecology 26, 297–302.

DiCiccio, T.J., Efron, B., 1996. Bootstrap confidence intervals. Statistical Science 11, 189 – 228. URL: https://doi.org/10.1214/ss/1032280214, doi:10.1214/ss/1032280214.

Dobriban, E., Sheng, Y., 2021. Distributed linear regression by averaging .

Dong, S., Wang, P., Abbas, K., 2021. A survey on deep learning and its applications. Computer Science Review 40, 100379.

Dong, Z., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K., 2019. Hawq: Hessian aware quantization of neural networks with mixed-precision, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 293–302.

Duan, W., Gu, X., Wen, M., Ji, Y., Ge, J., Zhang, G., 2022. Resource management for intelligent vehicular edge computing networks. IEEE Transactions on Intelligent Transportation Systems 23, 9797–9808. doi:10.1109/TITS.2021.3114957.

Efron, B., 1983. Estimating the error rate of a prediction rule: improvement on cross-validation. Journal of the American statistical association 78, 316–331.

Evci, U., Gale, T., Menick, J., Castro, P.S., Elsen, E., 2020. Rigging the lottery: Making all tickets winners, in: International Conference on Machine Learning, PMLR. pp. 2943–2952.

Feeney, L.M., Nilsson, M., 2001. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, in: Proceedings IEEE INFOCOM 2001. Conference on computer communications. Twentieth annual joint conference of the IEEE computer and communications society (Cat. No. 01CH37213), IEEE. pp. 1548–1557.

Feurer, M., Springenberg, J.T., Hutter, F., 2014. Using meta-learning to initialize bayesian optimization of hyperparameters., in: MetaSel@ ECAI, pp. 3–10.

Frankle, J., Carbin, M., 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 .

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., Smola, A., 2006. A kernel method for the two-sample-problem. Advances in neural information processing systems 19.

Guo, P., Hu, W., 2018. Potluck: Cross-application approximate deduplication for computation-intensive mobile applications, in: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 271–284.

Haddadpour, F., Mahdavi, M., 2019. On the convergence of local descent methods in federated learning. CoRR abs/1910.14425. URL: http://arxiv.org/abs/1910.14425, arXiv:1910.14425.

Han, J., Liu, Q., 2016. Bootstrap model aggregation for distributed statistical learning. Advances in Neural Information Processing Systems 29.

Han, P., Wang, S., Leung, K.K., 2020. Adaptive gradient sparsification for efficient federated learning: An online learning approach, in: 2020 IEEE 40th international conference on distributed computing systems (ICDCS), IEEE. pp. 300–310.

Han, S., Pool, J., Tran, J., Dally, W., 2015. Learning both weights and connections for efficient neural network. Advances in neural information processing systems 28.

Harth, N., Anagnostopoulos, C., 2018. Edge-centric efficient regression analytics, in: 2018 IEEE International Conference on Edge Computing (EDGE), pp. 93–100. doi:10.1109/EDGE.2018.00020.

Hasani, S., Ghaderi, F., Hasan, S., Thirumuruganathan, S., Asudeh, A., Koudas, N., Das, G., 2019. Approxml: efficient approximate ad-hoc ml models through materialization and reuse. Proc. VLDB Endow. 12, 1906–1909. URL: https://doi.org/10.14778/3352063.3352096, doi:10.14778/3352063.3352096.

Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H., 2009. The elements of statistical learning: data mining, inference, and prediction. volume 2. Springer.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: IEEE CVPR, pp. 770–778.

He, Y., Zhang, X., Sun, J., 2017. Channel pruning for accelerating very deep neural networks, in: IEEE ICCV, pp. 1389–1397.

Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 .

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., Peste, A., 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. J. Mach. Learn. Res. 22, 1–124.

Huang, T., Liu, S., Shen, L., He, F., Lin, W., Tao, D., 2022a. Achieving personalized federated learning with sparse local models. arXiv preprint arXiv:2201.11380 .

Huang, T., Shen, L., Sun, Y., Lin, W., Tao, D., 2022b. Fusion of global and local knowledge for personalized federated learning. Transactions on Machine Learning Research .

Idelbayev, Y., Carreira-Perpinán, M.A., 2020. Low-rank compression of neural nets: Learning the rank of each layer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8049–8059.

Idelbayev, Y., Carreira-Perpiñán, M.Á., 2021. Optimal selection of matrix shape and decomposition scheme for neural network compression, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 3250–3254.

ISACA, 2023. Practical data security and privacy for gdpr and ccpa. https://www.isaca.org.

Isik, B., Pase, F., Gunduz, D., Weissman, T., Michele, Z., 2023. Sparse random networks for communication-efficient federated learning, in: The Eleventh International Conference on Learning Representations.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D., 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2704–2713.

Jastrzebski, S., Arpit, D., Astrand, O., Kerg, G.B., Wang, H., Xiong, C., Socher, R., Cho, K., Geras, K.J., 2021. Catastrophic fisher explosion: Early phase fisher matrix impacts generalization, in: International Conference on Machine Learning, PMLR. pp. 4772–4784.

Jiang, P., Agrawal, G., 2018. A linear speedup analysis of distributed deep learning with sparse and quantized communication. NeurIPS'18 31.

Jiang, X., Borcea, C., 2023. Complement sparsification: Low-overhead model pruning for federated learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 8087–8095.

Jiang, Y., Wang, S., Valls, V., Ko, B.J., Lee, W.H., Leung, K.K., Tassiulas, L., 2022. Model pruning enables efficient federated learning on edge devices. IEEE TNNLS .

Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al., 2021. Advances and open problems in federated learning. Foundations and trends® in machine learning 14, 1–210.

Khan, W.Z., Ahmed, E., Hakak, S., Yaqoob, I., Ahmed, A., 2019. Edge computing: A survey. Future Generation Computer Systems 97, 219–235.

Ko, Y., 2012. A study of term weighting schemes using class information for text classification, in: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, pp. 1029–1030.

Kolomvatsos, K., Anagnostopoulos, C., 2022. A proactive statistical model supporting services and tasks management in pervasive applications. IEEE Transactions on Network and Service Management 10.1109/TNSM.2022.3161663, 1–1.

Kolomvatsos, K., Anagnostopoulos, C., Koziri, M., Loukopoulos, T., 2020. Proactive time-optimized data synopsis management at the edge. IEEE Transactions on Knowledge and Data Engineering , 1–1doi:10.1109/TKDE.2020.3021377.

Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D., 2016. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 .

Konečnỳ, J., Richtárik, P., 2018. Randomized distributed mean estimation: Accuracy vs. communication. Frontiers in Applied Mathematics and Statistics 4, 62.

Krizhevsky, A., 2014. One weird trick for parallelizing convolutional neural networks. CoRR abs/1404.5997. URL: http://arxiv.org/abs/1404.5997, arXiv:1404.5997.

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

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems 25.

Lai, F., Zhu, X., Madhyastha, H.V., Chowdhury, M., 2021. Oort: Efficient federated learning via guided participant selection, in: 15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21), pp. 19–35.

Lalitha, A., Shekhar, S., Javidi, T., Koushanfar, F., 2018. Fully decentralized federated learning, in: Third workshop on bayesian deep learning (NeurIPS).

Law, B., 2023. Consumer data privacy: Eu's gdpr vs. china's pipl. https://pro.bloomberglaw.com.

LeCun, Y., et al., 2015. Lenet-5, convolutional neural networks. URL: http://yann. lecun. com/exdb/lenet 20, 14.

Lee, J., Mtibaa, A., Mastorakis, S., 2019. A case for compute reuse in future edge systems: An empirical study, in: 2019 IEEE Globecom Workshops (GC Wkshps), pp. 1–6. doi:10.1109/GCWkshps45667.2019.9024587.

Lee, N., Ajanthan, T., Torr, P., 2018. Snip: Single-shot network pruning based on connection sensitivity, in: International Conference on Learning Representations.

Leite, R., Brazdil, P., 2005. Predicting relative performance of classifiers from samples, in: Proceedings of the 22nd international conference on machine learning, pp. 497–503.

Li, A., Sun, J., Zeng, X., Zhang, M., Li, H., Chen, Y., 2021a. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking, in: Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, pp. 42–55.

Li, C., Peng, J., Yuan, L., Wang, G., Liang, X., Lin, L., Chang, X., 2020a. Block-wisely supervised neural architecture search with knowledge distillation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1989–1998.

Li, C., Zeng, X., Zhang, M., Cao, Z., 2022. Pyramidfl: A fine-grained client selection framework for efficient federated learning, in: Proceedings of the 28th Annual International Conference on Mobile Computing And Networking, pp. 158–171.

Li, R., Ma, F., Jiang, W., Gao, J., 2019a. Online federated multitask learning, in: 2019 IEEE International Conference on Big Data (Big Data), IEEE. pp. 215–220.

Li, S., Liu, Z.Q., Chan, A.B., 2014. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network, in: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 482–489.

Li, T., Hu, S., Beirami, A., Smith, V., 2021b. Ditto: Fair and robust federated learning through personalization, in: International Conference on Machine Learning, PMLR. pp. 6357–6368.

Li, T., Li, J., Liu, Z., Zhang, C., 2020b. Few sample knowledge distillation for efficient network compression, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 14639–14647.

Li, T., Sahu, A.K., Talwalkar, A., Smith, V., 2020c. Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine 37, 50–60.

Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V., 2020d. Federated optimization in heterogeneous networks. PMLR 2, 429–450.

Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z., 2019b. On the convergence of fedavg on non-iid data, in: ICLR.

Li, Y., Zhang, Z., Liu, B., Yang, Z., Liu, Y., 2021c. Modeldiff: Testing-based dnn similarity comparison for model reuse detection. arXiv preprint arXiv:2106.08890 .

Liebenwein, L., Maalouf, A., Feldman, D., Rus, D., 2021. Compressing neural networks: Towards determining the optimal layer-wise decomposition. Advances in Neural Information Processing Systems 34, 5328–5344.

Lin, T., Stich, S.U., Barba Flores, L.F., Dmitriev, D., Jaggi, M., 2020. Dynamic model pruning with feedback, in: ICLR.

Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W.J., 2018. Deep Gradient Compression: Reducing the communication bandwidth for distributed training, in: ICLR.

Liu, J., Huang, J., Zhou, Y., Li, X., Ji, S., Xiong, H., Dou, D., 2022. From distributed machine learning to federated learning: A survey. Knowledge and Information Systems 64, 885–917.

Liu, S., Pan, S.J., Ho, Q., 2017. Distributed multi-task relationship learning, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 937–946.

Long, Q., Anagnostopoulos, C., Kolomvatsos, K., 2024a. Enhancing knowledge reusability: A distributed multitask machine learning approach. IEEE Transactions on Emerging Topics in Computing , 1–14doi:10.1109/TETC.2024.3390811.

Long, Q., Anagnostopoulos, C., Parambath, S.P., Bi, D., 2023. Feddip: Federated learning with extreme dynamic pruning and incremental regularization, in: 2023 IEEE International Conference on Data Mining (ICDM), IEEE. pp. 1187–1192.

Long, Q., Kolomvatsos, K., Anagnostopoulos, C., 2022. Knowledge reuse in edge computing environments. Journal of Network and Computer Applications 206, 103466. doi:https://doi.org/10.1016/j.jnca.2022.103466.

Long, Q., Wang, Q., Anagnostopoulos, C., Bi, D., 2024b. Decentralized personalized federated learning based on a conditional sparse-to-sparser scheme. arXiv preprint arXiv:2404.15943 .

Luo, B., Li, X., Wang, S., Huang, J., Tassiulas, L., 2021. Cost-effective federated learning design, in: IEEE INFOCOM 2021-IEEE Conference on Computer Communications, IEEE. pp. 1–10.

Luo, J., Deng, X., Zhang, H., Qi, H., 2018. Ultra-low latency service provision in edge computing, in: 2018 IEEE International Conference on Communications (ICC), pp. 1–6. doi:10.1109/ICC.2018.8422645.

Lym, S., Choukse, E., Zangeneh, S., Wen, W., Sanghavi, S., Erez, M., 2019. Prunetrain: fast neural network training by dynamic sparse model reconfiguration, in: SC'19, pp. 1–13.

MacKay, D.J.C., 2002. Information Theory, Inference and; Learning Algorithms. Cambridge University Press, USA.

Mao, Y., Zhang, J., Letaief, K.B., 2016. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE Journal on Selected Areas in Communications 34, 3590–3605.

McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A., 2017. Communication-efficient learning of deep networks from decentralized data, in: Artificial intelligence and statistics, PMLR. pp. 1273–1282.

Navia-Vazquez, A., Gutierrez-Gonzalez, D., Parrado-Hernández, E., Navarro-Abellan, J., 2006. Distributed support vector machines. IEEE Transactions on Neural Networks 17, 1091–1097.

Nour, B., Cherkaoui, S., Mlika, Z., 2022. Federated learning and proactive computation reuse at the edge of smart homes. IEEE Transactions on Network Science and Engineering 9, 3045–3056. doi:10.1109/TNSE.2021.3131246.

Nour, B., Mastorakis, S., Mtibaa, A., 2021. Whispering: Joint service offloading and computation reuse in cloud-edge networks, in: ICC 2021-IEEE International Conference on Communications, IEEE. pp. 1–6.

Pan, S.J., Yang, Q., 2009. A survey on transfer learning. IEEE Transactions on knowledge and data engineering 22, 1345–1359.

Rachwan, J., Zügner, D., Charpentier, B., Geisler, S., Ayle, M., Günnemann, S., 2022. Winning the lottery ahead of time: Efficient early network pruning, in: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (Eds.), Proceedings of the 39th International Conference on Machine Learning, PMLR. pp. 18293–18309. URL: https://proceedings.mlr.press/v162/rachwan22a.html.

Rahimi, A., Recht, B., 2007. Random features for large-scale kernel machines, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA. p. 1177–1184.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., Pedarsani, R., 2020. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization, in: International conference on artificial intelligence and statistics, PMLR. pp. 2021–2031.

Reynolds, D.A., 2009. Gaussian mixture models. Encyclopedia of biometrics 741, 659–663.

Ruan, X., Liu, Y., Yuan, C., Li, B., Hu, W., Li, Y., Maybank, S., 2020. Edp: An efficient decomposition and pruning scheme for convolutional neural network compression. IEEE Transactions on Neural Networks and Learning Systems 32, 4499–4513.

S., Z., B., G., A., D., J., H., Z., X., X., C.S., 2017. Cautionary tales on air-quality improvement in beijing., in: Proceedings. Mathematical, physical, and engineering sciences, Royal Society. p. 473(2205). doi:https://doi.org/10.1098/rspa.2017.0457.

Samikwa, E., Di Maio, A., Braun, T., 2022. Ares: Adaptive resource-aware split learning for internet of things. Computer Networks 218, 109380.

Sattler, F., Wiedemann, S., Müller, K.R., Samek, W., 2019. Robust and communication-efficient federated learning from non-iid data. IEEE transactions on neural networks and learning systems 31, 3400–3413.

Sharifani, K., Amini, M., 2023. Machine learning and deep learning: A review of methods and applications. World Information Technology and Engineering Journal 10, 3897–3904.

Shi, S., Zhao, K., Wang, Q., Tang, Z., Chu, X., 2019. A convergence analysis of distributed sgd with communication-efficient gradient sparsification., in: IJCAI, pp. 3411–3417.

Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016. Edge computing: Vision and challenges. IEEE internet of things journal 3, 637–646.

Shi, Y., Shen, L., Wei, K., Sun, Y., Yuan, B., Wang, X., Tao, D., 2023. Improving the model consistency of decentralized federated learning, in: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (Eds.), Proceedings of the 40th International Conference on Machine Learning, PMLR. pp. 31269–31291. URL: https://proceedings.mlr.press/v202/shi23d.html.

Shui, C., Abbasi, M., Robitaille, L.E., Wang, B., Gagné, C., 2019. A principled approach for learning task similarity in multitask learning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press. p. 3446–3452.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations (ICLR 2015), Computational and Biological Learning Society.

Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.S., 2017. Federated multi-task learning. Advances in neural information processing systems 30.

Sriperumbudur, B.K., Gretton, A., Fukumizu, K., Schölkopf, B., Lanckriet, G.R., 2010. Hilbert space embeddings and metrics on probability measures. J. Mach. Learn. Res. 11, 1517–1561.

Strom, N., 2015. Scalable distributed dnn training using commodity gpu cloud computing, in: 16th Intl Conf Speech Comm. Assoc.

Sudharsan, B., Breslin, J.G., Ali, M.I., 2020. Edge2train: A framework to train machine learning models (svms) on resource-constrained iot edge devices, in: Proceedings of the 10th International Conference on the Internet of Things, pp. 1–8.

Sun, J., Chen, T., Giannakis, G., Yang, Z., 2019. Communication-efficient distributed learning via lazily aggregated quantized gradients. Advances in Neural Information Processing Systems 32.

Sun, T., Li, D., Wang, B., 2022. Decentralized federated averaging. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 4289–4301.

Swaminathan, S., Garg, D., Kannan, R., Andres, F., 2020. Sparse low rank factorization for deep neural network compression. Neurocomputing 398, 185–196.

Tan, A.Z., Yu, H., Cui, L., Yang, Q., 2023. Towards personalized federated learning. IEEE Transactions on Neural Networks and Learning Systems 34, 9587–9603. doi:10.1109/TNNLS.2022.3160699.

Tang, Z., Shi, S., Li, B., Chu, X., 2022. Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication. IEEE Transactions on Parallel and Distributed Systems 34, 909–922.

Tang, Z., Shi, S., Wang, W., Li, B., Chu, X., 2020. Communication-efficient distributed deep learning: A comprehensive survey. arXiv preprint arXiv:2003.06307 .

Thapa, C., Arachchige, P.C.M., Camtepe, S., Sun, L., 2022. Splitfed: When federated learning meets split learning, in: AAAI, pp. 8485–8493.

Thrun, S., O'Sullivan, J., 1998. Clustering learning tasks and the selective cross-task transfer of knowledge, in: Learning to learn. Springer, pp. 235–257.

Ti, N.T., Le, L.B., 2017. Computation offloading leveraging computing resources from edge cloud and mobile peers, in: 2017 IEEE International Conference on Communications (ICC), pp. 1–6. doi:10.1109/ICC.2017.7997138.

Tolstikhin, I., Sriperumbudur, B.K., Schölkopf, B., 2016. Minimax estimation of maximum mean discrepancy with radial kernels, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA. p. 1938–1946.

Tsai, D., Yang, R., 2005. An eigenvalue-based similarity measure and its application in defect detection. Image Vis. Comput. 23, 1094–1101. URL: https://doi.org/10.1016/j.imavis.2005.07.014, doi:10.1016/j.imavis.2005.07.014.

Tschandl, P., Rosendahl, C., Kittler, H., 2018. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Scientific data 5, 1–9.

Vanschoren, J., 2018. Meta-learning: A survey. arXiv preprint arXiv:1810.03548 .

Vepakomma, P., Gupta, O., Swedish, T., Raskar, R., 2018. Split learning for health: Distributed deep learning without sharing raw patient data. arXiv preprint arXiv:1812.00564 .

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., Rellermeyer, J.S., 2020. A survey on distributed machine learning. Acm computing surveys (csur) 53, 1–33.

Viering, T., Loog, M., 2022. The shape of learning curves: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 7799–7819.

Wan, S., Lu, J., Fan, P., Shao, Y., Peng, C., Letaief, K.B., 2021a. Convergence analysis and system design for federated learning over wireless networks. IEEE JSAC 39, 3622–3639. doi:10.1109/JSAC.2021.3118351.

Wan, S., Lu, J., Fan, P., Shao, Y., Peng, C., Letaief, K.B., 2021b. Convergence analysis and system design for federated learning over wireless networks. IEEE Journal on Selected Areas in Communications 39, 3622–3639.

Wang, D., Shen, L., Luo, Y., Hu, H., Su, K., Wen, Y., Tao, D., 2023. Fedabc: targeting fair competition in personalized federated learning, in: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence, AAAI Press. URL: https://doi.org/10.1609/aaai.v37i8.26203, doi:10.1609/aaai.v37i8.26203.

Wang, H., Qin, C., Zhang, Y., Fu, Y., 2021. Neural pruning via growing regularization, in: ICLR.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., Khazaeni, Y., 2020. Federated learning with matched averaging, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=BkluqlSFDS.

Wang, H.P., Stich, S., He, Y., Fritz, M., 2022a. Progfed: Effective, communication, and computation efficient federated learning by progressive training, in: International Conference on Machine Learning, PMLR. pp. 23034–23054.

Wang, J., Kolar, M., Srerbo, N., 2016. Distributed multi-task learning, in: Artificial intelligence and statistics, PMLR. pp. 751–760.

Wang, J., Yang, X., Cui, S., Che, L., Lyu, L., Xu, D.D., Ma, F., 2024. Towards personalized federated learning via heterogeneous model reassembly. Advances in Neural Information Processing Systems 36.

Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., Cheng, J., 2018a. Two-step quantization for low-bit neural networks, in: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp. 4376–4384.

Wang, R., Yan, J., Wu, D., Wang, H., Yang, Q., 2018b. Knowledge-centric edge computing based on virtualized d2d communication systems. IEEE Communications Magazine 56, 32–38. doi:10.1109/MCOM.2018.1700876.

Wang, Y., Lin, L., Chen, J., 2022b. Communication-efficient adaptive federated learning, in: International Conference on Machine Learning, PMLR. pp. 22802–22838.

Wangni, J., Wang, J., Liu, J., Zhang, T., 2018. Gradient sparsification for communication-efficient distributed optimization. Advances in Neural Information Processing Systems 31.

Weiss, K., Khoshgoftaar, T.M., Wang, D., 2016. A survey of transfer learning. Journal of Big data 3, 1–40.

Wu, C., Wu, F., Lyu, L., Huang, Y., Xie, X., 2022. Communication-efficient federated learning via knowledge distillation. Nature communications 13, 2032.

Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J., 2016. Quantized convolutional neural networks for mobile devices, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4820–4828.

Wu, W., Li, M., Qu, K., Zhou, C., Shen, X., Zhuang, W., Li, X., Shi, W., 2023. Split learning over wireless networks: Parallel design and resource management. IEEE Journal on Selected Areas in Communications 41, 1051–1066.

Wu, X.Z., Xu, W., Liu, S., Zhou, Z.H., 2021. Model reuse with reduced kernel mean embedding specification. IEEE Transactions on Knowledge and Data Engineering , 1–1doi:10.1109/TKDE.2021.3086619.

Xiao, H., Rasul, K., Vollgraf, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 .

Xu, Y., Liao, Y., Xu, H., Ma, Z., Wang, L., Liu, J., 2022. Adaptive control of local updating and model compression for efficient federated learning. IEEE Transactions on Mobile Computing 22, 5675–5689.

Yan, G., Wang, H., Yuan, X., Li, J., 2023. Criticalfl: A critical learning periods augmented client selection framework for efficient federated learning, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2898–2907.

Yang, H., Tang, M., Wen, W., Yan, F., Hu, D., Li, A., Li, H., Chen, Y., 2020. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp. 678–679.

Yang, Z., Chen, M., Saad, W., Hong, C.S., Shikh-Bahaei, M., 2021. Energy efficient federated learning over wireless communication networks. IEEE Transactions on Wireless Communications 20, 1935–1949. doi:10.1109/TWC.2020.3037554.

Yao, X., Huang, C., Sun, L., 2018. Two-stream federated learning: Reduce the communication costs, in: 2018 IEEE Visual Communications and Image Processing (VCIP), pp. 1–4. doi:10.1109/VCIP.2018.8698609.

Yuan, L., Wang, Z., Sun, L., Yu, P.S., Brinton, C.G., 2024. Decentralized federated learning: A survey and perspective. IEEE Internet of Things Journal , 1–1doi:10.1109/JIOT.2024.3407584.

Zhang, K., Mao, Y., Leng, S., Maharjan, S., Zhang, Y., 2017. Optimal delay constrained offloading for vehicular edge computing networks, in: 2017 IEEE International Conference on Communications (ICC), pp. 1–6. doi:10.1109/ICC.2017.7997360.

Zhang, L., Bao, C., Ma, K., 2021a. Self-distillation: Towards efficient and compact neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 44, 4388–4403.

Zhang, M., Sapra, K., Fidler, S., Yeung, S., Alvarez, J.M., 2021b. Personalized federated learning with first order model optimization, in: International Conference on Learning Representations.

Zhang, T., Ye, S., Zhang, K., Tang, J., Wen, W., Fardad, M., Wang, Y., 2018. A systematic dnn weight pruning framework using alternating direction method of multipliers, in: ECCV, pp. 184–199.

Zhang, Y., Yang, Q., 2021. A survey on multi-task learning. IEEE Transactions on Knowledge and Data Engineering , 1–1doi:10.1109/TKDE.2021.3070203.

Zhang, Y., Yeung, D.Y., 2010. A convex formulation for learning task relationships in multi-task learning, in: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, AUAI Press, Arlington, Virginia, USA. p. 733–742.

Zhao, H., Li, B., Li, Z., Richtárik, P., Chi, Y., 2022. Beer: Fast $o(1/t)$ rate for decentralized nonconvex optimization with communication compression. Advances in Neural Information Processing Systems 35, 31653–31667.

Zhao, P., Cai, L.W., Zhou, Z.H., 2020. Handling concept drift via model reuse. Machine Learning 109, 533–568.

Zhou, X., Zhao, J., Han, H., Guet, C., 2022. Joint optimization of energy consumption and completion time in federated learning, in: 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), IEEE. pp. 1005–1017.

Zhou, Z.H., 2016. Learnware: on the future of machine learning. Frontiers Comput. Sci. 10, 589–590.

Zhu, M., Gupta, S., 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878 .

Zhu, X., Gong, S., et al., 2018. Knowledge distillation by on-the-fly native ensemble. Advances in neural information processing systems 31.

Zhuang, W., Chen, C., Lyu, L., 2023. When foundation model meets federated learning: Motivations, challenges, and future directions. arXiv preprint arXiv:2306.15546 .